

1

Welcome to Codewars I

1 points

Introduction

Now is the time to prove that you have what it takes to be a true warrior. Now is the time to start coding and having fun! As a rite of passage, you must create your very first program in this journey. However, we are feeling generous today, so we will not ask you to do something too difficult. For now you only have to write a program that takes no input and returns the string "Welcome to Codewars 2026!".

Exercise

Write a program that outputs the string "Welcome to Codewars 2026!".

Input

None.

Output

The string "Welcome to Codewars 2026!".

Example 1

Input

No input for this problem.

Output

```
Welcome to Codewars 2026!
```



<CODEWARS2026/>
LEÓN

Solutions

Python

```
def main():  
    print("Welcome to Codewars 2026!")  
if __name__ == "__main__":  
    main()
```

C++

```
#include <iostream>  
int main(void) {  
    std::cout << "Welcome to Codewars 2026!" << std::endl;  
    return 0;  
}
```

Java

```
class Welcome {  
    public static void main(String[] args) {  
        System.out.println("Welcome to Codewars 2026!");  
    }  
}
```

2

Welcome to Codewars II

2 points

Introduction

Handling input will also be important for the challenges you will face today. Now, you have to extend your first program a bit, so it takes a team name (like the name of your team, but it can be any arbitrary string) and outputs the string *Welcome to Codewars 2026, TeamName!*, where *TeamName* is the name of the team you input.

Exercise

Write a program that outputs the string *Welcome to Codewars 2026, TeamName!* where *TeamName* is the name of the team you input.

Input

A string representing a name of a team.

Output

The string *Welcome to Codewars 2026, TeamName!* where *TeamName* is the name of the team you input.

Example 1

Input

```
Trambolikos
```

Output

```
Welcome to Codewars 2026, Trambolikos!
```

Example 2

Input

Los top del Mundo

Output

Welcome to Codewars 2026, Los top del Mundo!

Solutions

Python

```
def main():  
    team = input()  
    print("Welcome to Codewars 2026, " + team + "!")  
if __name__ == "__main__":  
    main()
```

C++

```
#include <iostream>  
#include <string>  
int main(void) {  
    std::string team;  
    std::getline(std::cin, team);  
    std::cout << "Welcome to Codewars 2026, " << team << "!" << std::endl;  
    return 0;  
}
```

Java

```
import java.util.*;
```




<CODEWARS2026/>
LEÓN

```
class Welcome2 {  
    public static void main(String[] args) {  
        Scanner scanner = new Scanner(System.in);  
        String teamName = scanner.nextLine().trim();  
        System.out.println("Welcome to Codewars 2026, " + teamName+"!");  
    }  
}
```

<let's code/>



3

Reverse DNA

4 points

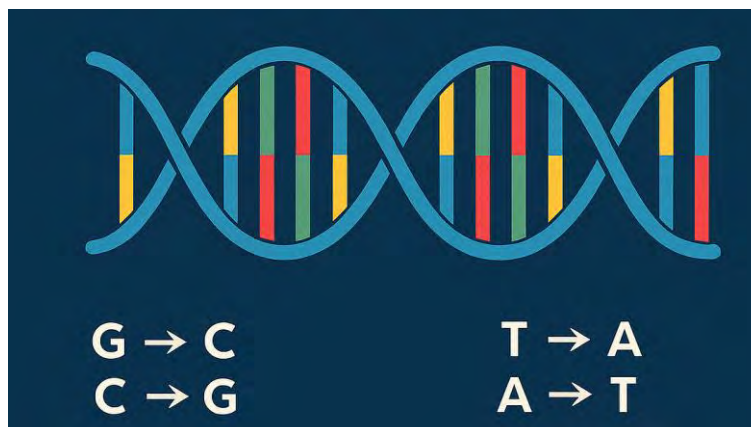
Introduction

You have been recently hired by a pharmaceutical company and are understanding the basics on heredability. One of your first tasks in the company is to understand that DNA (deoxyribonucleic acid) is the molecule that encapsulates all the information of the life species.

The encoding is done through four nucleotides: Adenine, Thymine, Cytosine and Guanine that are usually abbreviated as A, T, C, G. In each position of one DNA strand, we can have just one of these nucleotides and the paired strand codifies the opposite base (A links with T, C with G) so the molecule encodes two bits per base pair.

Therefore for each base, their corresponding pair is:

G → C
C → G
T → A
A → T



The complement of a DNA sequence is the sequence you get by replacing each base by its pair.

An equivalent protein called "reverse complement", is obtained if you find the complement of a sequence and then reverse it.

Exercise

You are asked to write a program that finds the reverse complement of a DNA sequence.

Input

The input of the program will be a list of sequences of nucleotide bases, each in one line, ending with a line with a single dot ('.'). As it's your first project, nucleotide sequences will always be shorter than 100 nucleotides (expressed as characters). An example of such input would be:

```
AAAACCCGGT
GCGCTTA
.
```

Output

Given the corresponding pairs of each nucleotide, your program will have to write the "reverse complement" of each nucleotide sequence in the list (in the same order). For the previous input, the expected output would be:

```
ACCGGGTTTT
TAAGCGC
```

Example 1

Input

```
A
```



<CODEWARS2026/> LEÓN

```
AC
ACG
ACGT
ACGAAGCGATAGCTGACTAGCTAGCATGCTAATCTAGA
.
```

Output

```
T
GT
CGT
ACGT
TCTAGATTAGCATGCTAGCTAGTCAGCTATCGCTTCGT
```

Example 2

Input

```
AAACACG
ACGTACGAAGCGATAGCTGACTAGCTAGCATGCTAATCTAGAACACG
.
```

Output

```
CGTGTTT
CGTGTTCTAGATTAGCATGCTAGCTAGTCAGCTATCGCTTCGTACGT
```

Solutions

Python

```
#!/usr/bin/env python3
import sys
# Tabla de complementos DNA
COMP = str.maketrans({
    'A': 'T',
    'T': 'A',
    'C': 'G',
```




<CODEWARS2026/> LEÓN

```
'G': 'C',
})
def reverse_complement(seq: str) -> str:
    # Complementa y revierte
    return seq.translate(COMP[::-1])
def main():
    out_lines = []
    for line in sys.stdin:
        s = line.strip()
        if not s:
            # Si hubiera líneas vacías, las ignoramos (no se especifican,
            # pero no molesta)
            continue
        if s == '.':
            break
        out_lines.append(reverse_complement(s))
    sys.stdout.write("\n".join(out_lines))
    if out_lines:
        sys.stdout.write("\n")
if __name__ == "__main__":
    main()
```

C++

```
#include <iostream>
#include <string>
#include <array>
#include <stdexcept>
static inline char complement(char b) {
    switch (b) {
```



<CODEWARS2026/>
LEÓN

```
        case 'A': return 'T';
        case 'T': return 'A';
        case 'C': return 'G';
        case 'G': return 'C';
        default:
            throw std::runtime_error(std::string("Base invalida: ") + b);
    }
}

int main() {
    std::ios::sync_with_stdio(false);
    std::cin.tie(nullptr);
    std::string s;
    while (std::getline(std::cin, s)) {
        // Por si el input viene con CRLF (Windows)
        if (!s.empty() && s.back() == '\\r') s.pop_back();
        if (s == ".") break;
        if (s.empty()) continue;
        // reverse + complement en una pasada
        for (int i = static_cast<int>(s.size()) - 1; i >= 0; --i) {
            std::cout << complement(s[i]);
        }
        std::cout << '\\n';
    }
    return 0;
}
```

Java

```
import java.io.BufferedReader;
import java.io.IOException;
```



<CODEWARS2026/>
LEÓN

```
import java.io.InputStreamReader;

public class dna {
    private static char complement(char base) {
        switch (base) {
            case 'A': return 'T';
            case 'T': return 'A';
            case 'C': return 'G';
            case 'G': return 'C';
            default:
                // Según el enunciado solo habrá A,C,G,T.
                // Si quieres hacerlo más robusto, puedes lanzar excepción
                // o devolver el mismo char.
                throw new IllegalArgumentException("Base inválida: " +
base);
        }
    }

    private static String reverseComplement(String seq) {
        StringBuilder sb = new StringBuilder(seq.length());
        // Recorremos desde el final: reverse + complement en una pasada
        for (int i = seq.length() - 1; i >= 0; i--) {
            sb.append(complement(seq.charAt(i)));
        }
        return sb.toString();
    }

    public static void main(String[] args) throws Exception {
        BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
        String line;
```



<CODEWARS2026/>
LEÓN

```
StringBuilder output = new StringBuilder();
while ((line = br.readLine()) != null) {
    line = line.trim();
    if (line.equals(".")) break;
    if (line.isEmpty()) continue; // por si acaso
    output.append(reverseComplement(line)).append('\n');
}
System.out.print(output.toString());
}
```

<let's code/>



4

The Galactic Alphabet Decoder

4 points



Introduction

Humanity has intercepted encrypted alien messages. Each letter in the message has been shifted forward by a constant value K in a cyclic alphabet ($A \rightarrow B \rightarrow C \dots \rightarrow Z \rightarrow A$). This means that after Z , the alphabet wraps around back to A .

Your task is to decode the message by shifting every letter backwards by K positions, restoring the original text. The backward shift is also cyclic: for example, shifting A backwards by 1 results in Z .

Exercise

Write a program that decodes alien encrypted messages.

Input

Your program will read two lines:

- An integer K ($0 \leq K \leq 25$)
- An uppercase string S (A-Z only)

Output

A single line containing the decoded message, obtained by shifting each letter of S backwards by K positions.

Example 1

Input

```
1  
BDF
```

Output

```
ACE
```

Example 2

Input

```
3  
KHOOR
```

Output

```
HELLO
```

Solutions

Python



<CODEWARS2026/> LEÓN

```
def main():
    K = int(input())
    S = input().strip()
    result = []
    for c in S:
        pos = ord(c) - ord('A')      # Convertimos letra → número 0-25
        new_pos = (pos - K) % 26      # Retrocedemos K posiciones con ciclo
        result.append(chr(new_pos + ord('A')))
    print("".join(result))
if __name__ == "__main__":
    main()
```

C++

```
#include <iostream>
#include <string>
int main() {
    int K;
    std::string S;
    std::cin >> K;          // Leemos el desplazamiento
    std::cin >> S;          // Leemos la cadena (sin espacios)
    std::string result;
    result.reserve(S.size());
    for (char c : S) {
        int pos = c - 'A';   // Convertimos letra → número
                                0-25
        int new_pos = (pos - K) % 26;    // Retrocedemos K posiciones
con ciclo
        // Aseguramos que el módulo sea positivo
        if (new_pos < 0) {
```




<CODEWARS2026/>
LEÓN

```
        new_pos += 26;
    }
    result.push_back(static_cast<char>(new_pos + 'A'));
}
std::cout << result << std::endl;
return 0;
}
```

Java

```
import java.util.Scanner;
public class galactic {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int K = scanner.nextInt();    // Leemos el desplazamiento
        String S = scanner.next();    // Leemos la cadena (sin espacios)
        StringBuilder result = new StringBuilder();
        for (char c : S.toCharArray()) {
            int pos = c - 'A';          // Convertimos letra →
número 0-25
            int newPos = (pos - K) % 26;    // Retrocedemos K
posiciones con ciclo
            // Aseguramos que el módulo sea positivo
            if (newPos < 0) {
                newPos += 26;
            }
            result.append((char)(newPos + 'A'));
        }
        System.out.println(result.toString());
        scanner.close();
    }
}
```

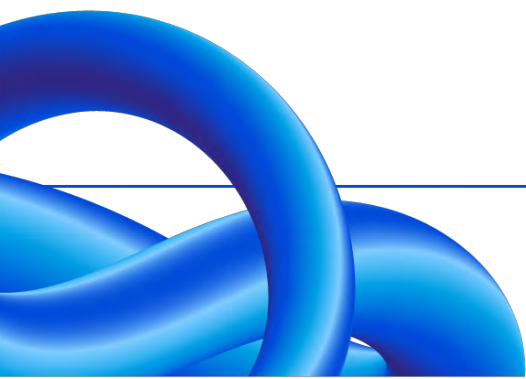
<let's code/>





<CODEWARS2026/>
LEÓN

}
}



<let's code/>



5

Mr. Burns infamous salary calculator

5 points

Introduction

Mr. Burns, the infamous owner of the Springfield Nuclear Power Plant, has decided to calculate the salaries of his employees using a custom formula. However, Mr. Burns wants to ensure that the calculations are performed accurately, following the proper operator precedence.



The formula he uses to calculate the salary is as follows:

$\text{salary} = \text{base} + \text{hours} \times \text{rate} - \text{deductions} / \text{tax} + (\text{bonus} - \text{penalty}) \times \text{multiplier}$

Where:

- base: The base salary of the employee.
- hours: The total number of hours worked by the employee.
- rate: The hourly wage.

- deductions: The total amount of deductions for the employee.
- tax: The tax rate applied to deductions.
- bonus: Any additional bonus the employee earns.
- penalty: Penalty amount for any misconduct or inefficiency.
- multiplier: A performance multiplier based on the employee's evaluation.

Exercise

Your task:

1. Write a program that allows Mr. Burns to input the values for base, hours, rate, deductions, tax, bonus, penalty, and multiplier.
2. Use the formula provided to calculate the employee's salary.
3. Print the final salary for Mr. Burns to review.

Remember! Each number must be used only once and all numbers must be greater than 0.

Take note that divisions will be always obtained with only 2 digit numbers.

Input

The input will consist in 8 numbers, each one entered on one line, that corresponds to each of the values used to calculate the salary:

```
2000
40
25
300
15
500
```



<CODEWARS2026/> LEÓN

100

1.2

Output

The output will show the calculated salary, with 2 digits:

3460.00

or

Please enter valid numeric values.

or

Division by zero occurred. Check the tax rate.

Example 1

Input

2000

40

25

300

15

500

100

1.2

Output

3460.00

Example 2

Input

2109

111

11

300



<CODEWARS2026/> LEÓN

```
10
240
80
1.3
```

Output

```
3508.00
```

Solutions

Python

```
def calculate_salary(base, hours, rate, deductions, tax, bonus, penalty,
multiplier):
    """Calculates the employee's salary based on the provided formula."""
    try:
        # Apply the formula
        salary = base + (hours * rate) - (deductions / tax) + ((bonus -
penalty) * multiplier)
        return salary
    except ZeroDivisionError:
        return "Error: Division by zero occurred. Check the tax rate."
def main():
    # Get user inputs
    try:
        base = float(input())
        hours = float(input())
        rate = float(input())
        deductions = float(input())
        tax = float(input())
        bonus = float(input())
        penalty = float(input())
```



<CODEWARS2026/>
LEÓN

```
multiplier = float(input())
if base<0 or hours<0 or rate <0 or deductions<0 or tax <0 or
bonus<0 or penalty<0 or multiplier<0:
    raise ValueError("")
# Calculate salary
salary = calculate_salary(base, hours, rate, deductions, tax,
bonus, penalty, multiplier)
# Display result
print(f"{salary:.2f}" if isinstance(salary, float) else salary)
except ValueError:
    print("Please enter valid numeric values.")
if __name__ == "__main__":
    main()
```

C++

```
#include <iostream>
#include <iomanip>
#include <stdexcept>
double calculate_salary(double base, double hours, double rate, double
deductions, double tax, double bonus, double penalty, double multiplier) {
    // Aplica la fórmula
    if (tax == 0) throw std::runtime_error("Error: Division by zero
occurred. Check the tax rate.");
    return base + (hours * rate) - (deductions / tax) + ((bonus - penalty)
* multiplier);
}
int main() {
    double base, hours, rate, deductions, tax, bonus, penalty, multiplier;
    try {
```

```
std::cin >> base >> hours >> rate >> deductions >> tax >> bonus >>
penalty >> multiplier;
if (base < 0 || hours < 0 || rate < 0 || deductions < 0 || tax < 0
|| bonus < 0 || penalty < 0 || multiplier < 0) {
    throw std::invalid_argument("");
}
double salary = calculate_salary(base, hours, rate, deductions,
tax, bonus, penalty, multiplier);
std::cout << std::fixed << std::setprecision(2) << salary <<
std::endl;
} catch (const std::invalid_argument&) {
    std::cout << "Please enter valid numeric values." << std::endl;
} catch (const std::runtime_error& e) {
    std::cout << e.what() << std::endl;
} catch (...) {
    std::cout << "Please enter valid numeric values." << std::endl;
}
return 0;
}
```

Java

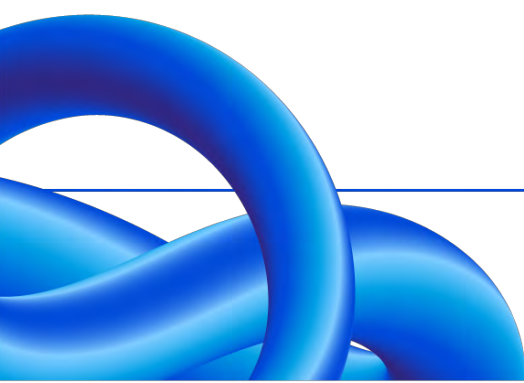
```
import java.util.Scanner;
public class Mr_Burns_infamous_salary_calculator {
    public static double calculateSalary(double base, double hours, double
rate, double deductions, double tax, double bonus, double penalty, double
multiplier) throws ArithmeticException {
        if (tax == 0) throw new ArithmeticException("Error: Division by
zero occurred. Check the tax rate.");
    }
}
```

```
        return base + (hours * rate) - (deductions / tax) + ((bonus -  
penalty) * multiplier);  
    }  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        try {  
            double base = Double.parseDouble(sc.nextLine());  
            double hours = Double.parseDouble(sc.nextLine());  
            double rate = Double.parseDouble(sc.nextLine());  
            double deductions = Double.parseDouble(sc.nextLine());  
            double tax = Double.parseDouble(sc.nextLine());  
            double bonus = Double.parseDouble(sc.nextLine());  
            double penalty = Double.parseDouble(sc.nextLine());  
            double multiplier = Double.parseDouble(sc.nextLine());  
            if (base < 0 || hours < 0 || rate < 0 || deductions < 0 || tax  
< 0 || bonus < 0 || penalty < 0 || multiplier < 0) {  
                throw new IllegalArgumentException();  
            }  
            double salary = calculateSalary(base, hours, rate, deductions,  
tax, bonus, penalty, multiplier);  
            System.out.printf("%.2f\n", salary);  
        } catch (IllegalArgumentException e) {  
            System.out.println("Please enter valid numeric values.");  
        } catch (ArithmeticException e) {  
            System.out.println("Error: Division by zero occurred. Check  
the tax rate.");  
        }  
    }  
}
```




<CODEWARS2026/>
LEÓN

}



<let's code/>



6

Pokemon Battle Effectiveness Simulator

5 points



Introduction

Ash Ketchum is preparing for the Pokemon World Tournament, and he needs your help! To win against tough opponents, Ash needs to understand how effective his Pokemon's moves are against different types.

Your task is to create a Pokemon Battle Effectiveness Simulator. Use the following type-effectiveness table to determine how strong an attack of one type is against a Pokemon of another type.

The numbers represent the attack's effectiveness:

#	Attacker\ Defender	Fire	Water	Grass	Electric	Ground	Rock	Flying	Ice	Psychic	Dragon
1	Fire	1.0	0.5	2.0	1.0	1.0	0.5	1.0	2.0	1.0	0.5

2	Water	2.0	1.0	0.5	1.0	2.0	2.0	1.0	1.0	1.0	0.5
3	Grass	0.5	2.0	1.0	1.0	2.0	1.0	0.5	1.0	1.0	0.5
4	Electric	1.0	1.0	2.0	1.0	0.0	1.0	2.0	1.0	1.0	0.5
5	Groun d	2.0	1.0	0.5	2.0	1.0	2.0	0.0	1.0	1.0	1.0
6	Rock	2.0	1.0	2.0	1.0	0.5	1.0	1.0	1.0	1.0	1.0
7	Flying	1.0	1.0	2.0	0.5	1.0	0.5	1.0	2.0	1.0	1.0
8	Ice	0.5	0.5	2.0	1.0	2.0	1.0	1.0	0.5	1.0	2.0
9	Psychic	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
10	Dragon	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	2.0

And the following is the table of texts that our program will show:

Value	Text to be shown
2.0	SUPER EFFECTIVE
1.0	NEUTRAL
0.5	NOT VERY EFFECTIVE
0.0	HAS NO EFFECT

Exercise

Write a program that:

- Receive the user selection of an attacker's type and a defender's type.
- Uses the table to determine the attack's effectiveness.
- Prints whether the attack is SUPER EFFECTIVE, NEUTRAL, NOT VERY EFFECTIVE, or HAS NO EFFECT.

Input

The input will consist in two lines with two positive integer numbers that represent the attacker's and defender's pokemon types. Therefore if we receive two numbers 1 and 3, the Attacker will be "Fire" and the Defender "Grass".

Output

The output is a single line of text like:

The 6XXXX type attack against the YYYYY type is ZZZZZZ

Where XXXX and YYYYY are the name of the attacker and defender types (for instance Fire and Grass in our example) and ZZZZZ is the result, based on the number and the text from the previous tables.

Example 1

Input

```
1  
3
```

Output

```
The Fire type attack against the Grass type is SUPER EFFECTIVE!
```

Example 2

Input

```
4  
5
```

Output

```
The Electric type attack against the Ground type HAS NO EFFECT.
```

Example 3

Input

```
6  
2
```

Output

```
The Rock type attack against the Water type is NOT VERY EFFECTIVE.
```


Solutions

Python

```
def main():
    # Define Pokémon types
    types = [
        "Fire", "Water", "Grass", "Electric", "Ground",
        "Rock", "Flying", "Ice", "Psychic", "Dragon"
    ]

    # Effectiveness matrix: 2.0 (Super Effective), 1.0 (Neutral), 0.5 (Not
    # Very Effective), 0.0 (No Effect)
    effectiveness = [
        # Fire
        [1.0, 0.5, 2.0, 1.0, 1.0, 0.5, 1.0, 2.0, 1.0, 0.5], # Fire
        # Water
        [2.0, 1.0, 0.5, 1.0, 2.0, 2.0, 1.0, 1.0, 1.0, 0.5], # Water
        # Grass
        [0.5, 2.0, 1.0, 1.0, 2.0, 1.0, 0.5, 1.0, 1.0, 0.5], # Grass
        # Electric
        [1.0, 1.0, 2.0, 1.0, 0.0, 1.0, 2.0, 1.0, 1.0, 0.5], # Electric
        # Ground
        [2.0, 1.0, 0.5, 2.0, 1.0, 2.0, 0.0, 1.0, 1.0, 1.0], # Ground
        # Rock
        [2.0, 1.0, 2.0, 1.0, 0.5, 1.0, 1.0, 1.0, 1.0, 1.0], # Rock
        # Flying
        [1.0, 1.0, 2.0, 0.5, 1.0, 0.5, 1.0, 2.0, 1.0, 1.0], # Flying
    ]
```



<CODEWARS2026/>
LEÓN

```
[0.5, 0.5, 2.0, 1.0, 2.0, 1.0, 1.0, 0.5,
1.0, 2.0], # Ice
[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
1.0, 1.0], # Psychic
[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
1.0, 2.0] # Dragon
]

# Input: Prompt for a single line with attacker and defender indices
(e.g., "1 3")
try:
    attacker = int(input().strip())
    defender = int(input().strip())
    attacker -= 1 # Convert to 0-based index
    defender -= 1 # Convert to 0-based index
except Exception as e:
    # Handle invalid input format
    return

# Validate input
if attacker < 0 or attacker >= len(types) or defender < 0 or defender
>= len(types):
    return

# Calculate effectiveness
result = effectiveness[attacker][defender]
# Determine effectiveness description
if result == 2.0:
    effectiveness_description = "is SUPER EFFECTIVE!"
elif result == 1.0:
    effectiveness_description = "is neutral."
```



<CODEWARS2026/>
LEÓN

```
elif result == 0.5:
    effectiveness_description = "is NOT VERY EFFECTIVE."
elif result == 0.0:
    effectiveness_description = "HAS NO EFFECT."
else:
    effectiveness_description = "has an unknown effectiveness."
# Display the result
print(f"The {types[attacker]} type attack against the
{types[defender]} type {effectiveness_description}")
if __name__ == "__main__":
    main()
```

C++

```
#include <iostream>
#include <string>
#include <stdexcept>
int main() {
    // Define Pokémon types
    std::string types[] = {
        "Fire", "Water", "Grass", "Electric", "Ground",
        "Rock", "Flying", "Ice", "Psychic", "Dragon"
    };
    // Effectiveness matrix: 2.0 (Super Effective), 1.0 (Neutral), 0.5
    (Not Very Effective), 0.0 (No Effect)
    double effectiveness[10][10] = {
        // Fire    Water  Grass  Electric Ground Rock  Flying Ice
    Psychic Dragon
        {1.0,    0.5,    2.0,    1.0,    1.0,    0.5,    1.0,    2.0,
    1.0,    0.5}, // Fire
```



<CODEWARS2026/> LEÓN

```
        {2.0,    1.0,    0.5,    1.0,    2.0,    2.0,    1.0,    1.0,
1.0,    0.5}, // Water
        {0.5,    2.0,    1.0,    1.0,    2.0,    1.0,    0.5,    1.0,
1.0,    0.5}, // Grass
        {1.0,    1.0,    2.0,    1.0,    0.0,    1.0,    2.0,    1.0,
1.0,    0.5}, // Electric
        {2.0,    1.0,    0.5,    2.0,    1.0,    2.0,    0.0,    1.0,
1.0,    1.0}, // Ground
        {2.0,    1.0,    2.0,    1.0,    0.5,    1.0,    1.0,    1.0,
1.0,    1.0}, // Rock
        {1.0,    1.0,    2.0,    0.5,    1.0,    0.5,    1.0,    2.0,
1.0,    1.0}, // Flying
        {0.5,    0.5,    2.0,    1.0,    2.0,    1.0,    1.0,    0.5,
1.0,    2.0}, // Ice
        {1.0,    1.0,    1.0,    1.0,    1.0,    1.0,    1.0,    1.0,
1.0,    1.0}, // Psychic
        {1.0,    1.0,    1.0,    1.0,    1.0,    1.0,    1.0,    1.0,
1.0,    2.0} // Dragon
    };
    // Prompt for a single input
    int attacker, defender;
    try {
        std::cin >> attacker >> defender;
        attacker--; // Adjust to 0-based index
        defender--; // Adjust to 0-based index
    } catch (const std::exception& e) {
        std::cerr << "Invalid input format" << std::endl;
        return 1;
    }
```




<CODEWARS2026/>
LEÓN

```
}  
// Validate input  
if (attacker < 0 || attacker >= 10 || defender < 0 || defender >= 10)  
{  
    std::cerr << "Invalid type indices" << std::endl;  
    return 1;  
}  
// Calculate effectiveness  
double result = effectiveness[attacker][defender];  
// Determine effectiveness description  
std::string effectivenessDescription;  
if (result == 2.0) {  
    effectivenessDescription = "is SUPER EFFECTIVE!";  
} else if (result == 1.0) {  
    effectivenessDescription = "is neutral.";  
} else if (result == 0.5) {  
    effectivenessDescription = "is NOT VERY EFFECTIVE.";  
} else if (result == 0.0) {  
    effectivenessDescription = "HAS NO EFFECT.";  
} else {  
    effectivenessDescription = "has an unknown effectiveness.";  
}  
// Display the result  
std::cout << "The " << types[attacker] << " type attack against the "  
    << types[defender] << " type " << effectivenessDescription  
<< std::endl;  
    return 0;  
}
```

Java

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        // Define Pokémon types
        String[] types = {
            "Fire", "Water", "Grass", "Electric", "Ground",
            "Rock", "Flying", "Ice", "Psychic", "Dragon"
        };
        // Effectiveness matrix: 2.0 (Super Effective), 1.0 (Neutral), 0.5
        (Not Very Effective), 0.0 (No Effect)
        double[][] effectiveness = {
            // Fire    Water  Grass  Electric Ground Rock  Flying Ice
            Psychic Dragon
            {1.0,    0.5,    2.0,    1.0,    1.0,    0.5,    1.0,
2.0,    1.0,    0.5}, // Fire
            {2.0,    1.0,    0.5,    1.0,    2.0,    2.0,    1.0,
1.0,    1.0,    0.5}, // Water
            {0.5,    2.0,    1.0,    1.0,    2.0,    1.0,    0.5,
1.0,    1.0,    0.5}, // Grass
            {1.0,    1.0,    2.0,    1.0,    0.0,    1.0,    2.0,
1.0,    1.0,    0.5}, // Electric
            {2.0,    1.0,    0.5,    2.0,    1.0,    2.0,    0.0,
1.0,    1.0,    1.0}, // Ground
            {2.0,    1.0,    2.0,    1.0,    0.5,    1.0,    1.0,
1.0,    1.0,    1.0}, // Rock
            {1.0,    1.0,    2.0,    0.5,    1.0,    0.5,    1.0,
2.0,    1.0,    1.0}, // Flying
```



<CODEWARS2026/>
LEÓN

```
        {0.5,    0.5,    2.0,    1.0,    2.0,    1.0,    1.0,
0.5,    1.0,    2.0}, // Ice
        {1.0,    1.0,    1.0,    1.0,    1.0,    1.0,    1.0,
1.0,    1.0,    1.0}, // Psychic
        {1.0,    1.0,    1.0,    1.0,    1.0,    1.0,    1.0,
1.0,    1.0,    2.0} // Dragon
    };
    Scanner scanner = new Scanner(System.in);
    // Prompt for a single input
    // User enters attacker and defender types in a single line (e.g.,
    "1 3")
    int attacker, defender;
    try {
        // Parse the input into attacker and defender indices
        String input = scanner.nextLine();
        attacker = Integer.parseInt(input) -1;
        String input2 = scanner.nextLine();
        defender = Integer.parseInt(input2)-1;
    } catch (Exception e) {
        // Handle invalid input format
        scanner.close();
        return;
    }
    // Validate input
    if (attacker < 0 || attacker >= types.length || defender < 0 ||
defender >= types.length) {
        scanner.close();
        return;
    }
}
```



<CODEWARS2026/>
LEÓN

```
    }  
    // Calculate effectiveness  
    double result = effectiveness[attacker][defender];  
    // Determine effectiveness description  
    String effectivenessDescription;  
    if (result == 2.0) {  
        effectivenessDescription = "is SUPER EFFECTIVE!";  
    } else if (result == 1.0) {  
        effectivenessDescription = "is neutral.";  
    } else if (result == 0.5) {  
        effectivenessDescription = "is NOT VERY EFFECTIVE.";  
    } else if (result == 0.0) {  
        effectivenessDescription = "HAS NO EFFECT.";  
    } else {  
        effectivenessDescription = "has an unknown effectiveness.";  
    }  
    // Display the result  
    System.out.println("The " + types[attacker] + " type attack  
against the "  
        + types[defender] + " type " + effectivenessDescription);  
    scanner.close();  
}  
}
```


7

The Mysterious Pascal's Triangle

5 points

Introduction

Welcome, young mathematician! Deep in Numeroland, the land of Numbers, the wise wizard Pascal has hidden a magical triangle. This triangle holds secrets of patterns and powers, but only those who can decode its last row will unlock its mysteries!



The wizard Pascal didn't just make up the numbers in his triangle, he used a clever spell! Here's how you can conjure up the numbers yourself:

- The very top of the triangle always starts with a single 1. This is the tip of the wizard's hat!
- Every row starts and ends with 1. These are the magical borders that protect the triangle.
- Every other number inside the triangle is created by adding the two numbers directly above it—one from the left and one from the right. It's like each number is the child of two parents above it!

For example, to create the triangle of size 2:

```
1
1 1
1 2 1
```

See that 2 in the middle? It's made by adding the two 1s above it ($1 + 1 = 2$). Every number in the triangle (except the edges) is made this way!

Exercise

Your quest:

1. Input: The wizard will give you a number, N ($1 \leq N \leq 50$). This is the number of rows in Pascal's Triangle he wants you to explore.
2. Triangle Clues: For the first N rows (except the last), draw the triangle using only / and \ characters, so the triangle looks magical and mysterious.
3. The Final Secret: The last row of Pascal's Triangle is the key! Print each number in the last row, but with a twist: write each digit vertically, one below the other. For example, if you need to write 10, the "1" will be in the first line and the "0" in the second line. Are you ready to reveal the secrets of Pascal's Triangle and become a legend in the land of Numbers? Good luck, code wizard!

Input

The size of the Pascal's triangle with a single integer number lower than 50.

Output

If the number is less than 1, shout:

```
Error! Triangulo imposible.
```

If the number is greater than 50, warn:

```
Error! Triangulo demasiado grande.
```

The triangle of the specified size with the last row values in vertical format. Please note that the numbers for the last row values are padded with spaces to the right to have the same width

Important Note: Please Note that when a triangle is drawn, a blank line is added behind the triangle. This line is a white spaces line, as long as the previous line.

Example 1

Input

3

Output

```
  /\n /\n /\n /\n 1 3 3 1
```

Example 2

Input

5

Output

```
  /\n /\n /\n /\n /\n /\n 1 5 1 1 5 1
  0 0
```

Note: the last row for 5 is "1 5 10 10 5 1", so the digits of "10" are printed vertically.

Solutions

Python

```
def main():
    try:
        numRows = int(input())
    except Exception:
        print("Error! Triangulo imposible.")
        return
    if numRows < 1:
        print("Error! Triangulo imposible.")
        return
    if numRows > 50:
        print("Error! Triangulo demasiado grande.")
        return
    # Calculate only the last line of Pascal's Triangle, using Python's
    big integers
    row = [1] * (numRows + 1)
    for i in range(1, numRows):
        for j in range(i, 0, -1):
            row[j] += row[j - 1]
    # Print the n-1 first lines of Pascal's Triangle as '/' and '\\',
    centered
    for i in range(numRows):
        print(' ' * (numRows - i - 1), end='')
        print(''.join('/') if j % 2 == 0 else '\\' for j in range(i * 2 +
2)))
```




<CODEWARS2026/>
LEÓN

```
# Print the last line, each digit vertically, with spaces between
numbers
num_strs = [str(n) for n in row]
digit_pos = 0
while True:
    has_more_digits = False
    line = []
    for num_str in num_strs:
        if digit_pos < len(num_str):
            line.append(num_str[digit_pos])
            has_more_digits = True
        else:
            line.append(' ')
    print(' '.join(line))
    if not has_more_digits:
        break
    digit_pos += 1
if __name__ == "__main__":
    main()
```

C++

```
// PascalsTriangle.cpp : This file contains the 'main' function. Program
execution begins and ends there.
//
#include <iostream>
#include <vector>
#include <string>
int main()
{
```



<CODEWARS2026/> LEÓN

```
// We need to get an integer from the standard input
int numRows;
std::cin >> numRows;

// If the number is less than 1, we should show the message "Error!
Triangulo imposible."
if (numRows < 1) {
    std::cout << "Error! Triangulo imposible." << std::endl;
    return 0;
}

// If the number is greater than 50, we should show the message
"Error! Triangulo demasiado grande."
if (numRows > 50) {
    std::cout << "Error! Triangulo demasiado grande." << std::endl;
    return 0;
}

// We should calculate only the last line of Pascal's Triangle, using
64bit integers
std::vector<int64_t> row(numRows + 1, 1);
for (int i = 1; i < numRows; ++i) {
    for (int j = i; j > 0; --j) {
        row[j] += row[j - 1];
    }
}

// Print the n-1 first lines of Pascal's Triangle only as characters
'/' and '\' indicating the relationship of the values, centered along the
base
for (int i = 0; i < numRows; ++i) {
    // Print leading spaces
```



```
for (int s = 0; s < numRows - i - 1; ++s) {
    std::cout << ' ';
}
// Print the values with '/' and '\\'
for (int j = 0; j <= i*2 + 1; ++j) {
    std::cout << (j % 2 == 0 ? '/' : '\\') ;
}
std::cout << '\n';
}

// Print the last line of Pascal's Triangle. Each number should be
written in vertical, that is, each digit in it's own line whith one space
between different numbers.
for (int digitPos = 0; ; ++digitPos) {
    bool hasMoreDigits = false;
    for (int j = 0; j <= numRows; ++j) {
        std::string numStr = std::to_string(row[j]);
        if (digitPos < numStr.size()) {
            std::cout << numStr[digitPos];
            hasMoreDigits = true;
        } else {
            std::cout << ' ';
        }
        if (j < numRows) {
            std::cout << ' '; // space between numbers
        }
    }
    std::cout << '\n';
    if (!hasMoreDigits) {
```



<CODEWARS2026/>
LEÓN

```
        break;
    }
}
}
```

Java

```
import java.util.Scanner;
public class PascalsTriangle {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int numRows;
        try {
            numRows = Integer.parseInt(sc.nextLine().trim());
        } catch (Exception e) {
            System.out.println("Error! Triangulo imposible.");
            return;
        }
        if (numRows < 1) {
            System.out.println("Error! Triangulo imposible.");
            return;
        }
        if (numRows > 50) {
            System.out.println("Error! Triangulo demasiado grande.");
            return;
        }
        // Calcular solo la última fila del triángulo de Pascal
        java.math.BigInteger[] row = new java.math.BigInteger[numRows +
1];
```




<CODEWARS2026/>
LEÓN

```
        for (int i = 0; i <= numRows; i++) row[i] =
java.math.BigInteger.ONE;
        for (int i = 1; i < numRows; i++) {
            for (int j = i; j > 0; j--) {
                row[j] = row[j].add(row[j - 1]);
            }
        }
// Imprimir las n-1 primeras líneas con '/' y '\', centradas
for (int i = 0; i < numRows; i++) {
    for (int s = 0; s < numRows - i - 1; s++) System.out.print("
");
    for (int j = 0; j < i * 2 + 2; j++) System.out.print(j % 2 ==
0 ? "/" : "\\");
    System.out.println();
}
// Imprimir la última línea, cada dígito verticalmente, con
espacios entre números
String[] numStrs = new String[row.length];
int maxLen = 0;
for (int i = 0; i < row.length; i++) {
    numStrs[i] = row[i].toString();
    if (numStrs[i].length() > maxLen) maxLen =
numStrs[i].length();
}
int digitPos = 0;
while (true) {
    boolean hasMoreDigits = false;
    StringBuilder line = new StringBuilder();
```



<CODEWARS2026/>
LEÓN

```
for (int i = 0; i < numStrs.length; i++) {  
    if (digitPos < numStrs[i].length()) {  
        line.append(numStrs[i].charAt(digitPos));  
        hasMoreDigits = true;  
    } else {  
        line.append(" ");  
    }  
    if (i < numStrs.length - 1) line.append(" ");  
}  
System.out.println(line);  
if (!hasMoreDigits) break;  
digitPos++;  
}  
}
```

<let's code/>



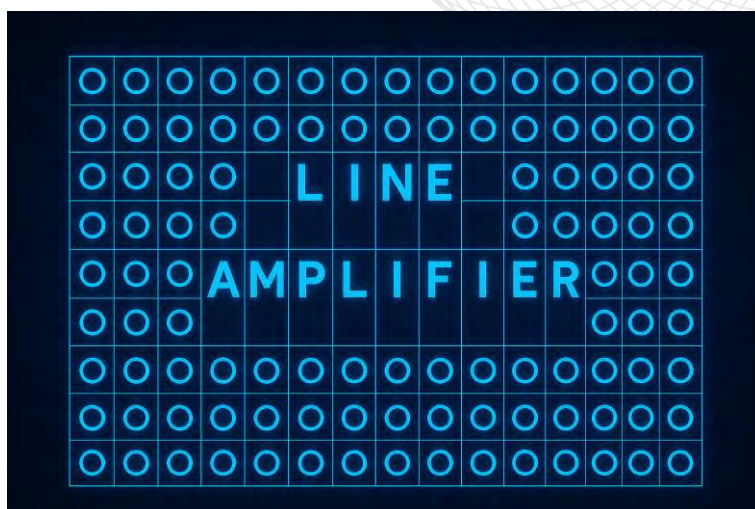


<CODEWARS2026/>
LEÓN

8

Line Amplifier

6 points



Introduction

You are requested to write a program that amplifies lines drawn through standard input with letters '8'. The program will receive a simulated 10x10 board thorough the standard input where different lines could be drawn suing the character 'X'. For instance:

```
X  
X  
X  
XXXX
```

In this case we have drawn letter 'L'. Note that 2 empty lines and 3 empty columns exist before the first 'X'.

You are asked to make kind of a "Bold" function so that the 'X's are surrounded by letters 'O'.

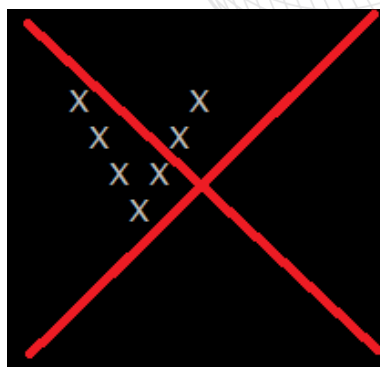
For instance with the previous input, the expected output is:

```
OOO  
OXO  
OXO  
OXOOOO  
OXXXXO  
OOOOOO
```

Note that the board is still 10x10 (10 lines and 10 columns).

Exercise

You are asked to write a program that amplifies anything that's written on the board. To make things easier, the drawings will only have 90° angles (nothing like the following):



Input

The program will read 10 lines (which will be the board). Note that there will always be at least one blank line and one blank column before the first 'X'. And an empty line and column after the last 'X'. This is to give space for the amplification

Output

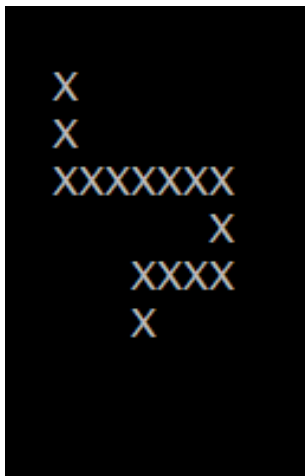
The program will write a 10 dash separator, and then the 10 lines board with the amplified drawing.

Example 1

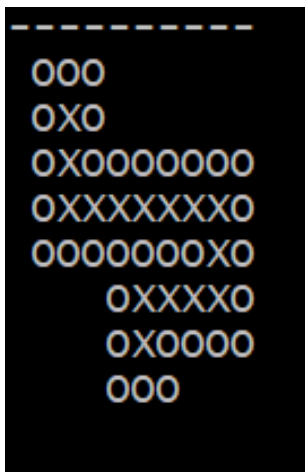
Input



<CODEWARS2026/> LEÓN



Output

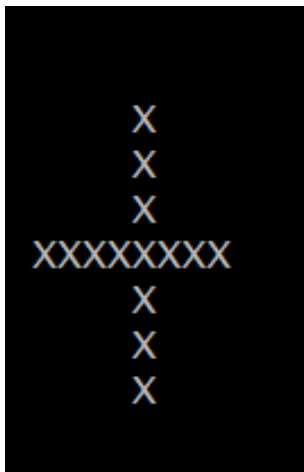


Example 2

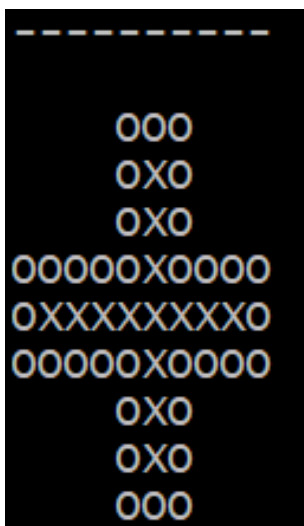
Input



<CODEWARS2026/>
LEÓN



Output



Solutions

Python

```
def ampliar_dibujo():  
    # Leer 10 líneas desde entrada estándar
```



<CODEWARS2026/> LEÓN

```
lineas = [input() for _ in range(10)]
print ("-----")
# Determinar dimensiones originales
filas = len(lineas)
columnas = max(len(linea) for linea in lineas)
# Crear matriz original con espacios
matriz = [list(linea.ljust(columnas)) for linea in lineas]
# Crear nueva matriz con borde adicional
nueva_filas = filas
nueva_columnas = columnas +1
nueva_matriz = [[' ' for _ in range(nueva_columnas)] for _ in
range(nueva_filas)]
# Copiar las X en la posición desplazada
for i in range(filas):
    for j in range(columnas):
        if matriz[i][j] == 'X':
            nueva_matriz[i][j] = 'X'
# Añadir 0 alrededor del bloque que contiene X
for i in range(nueva_filas):
    for j in range(nueva_columnas):
        if nueva_matriz[i][j] == 'X':
            # Marcar las posiciones vecinas como 0 si están vacías
            for di in [-1, 0, 1]:
                for dj in [-1, 0, 1]:
                    ni, nj = i + di, j + dj
                    if 0 <= ni < nueva_filas and 0 <= nj <
nueva_columnas:
                        if nueva_matriz[ni][nj] == ' ':

```




<CODEWARS2026/>
LEÓN

```
nueva_matriz[ni][nj] = '0'

# Imprimir el resultado
for fila in nueva_matriz:
    print(''.join(fila).rstrip())
# Ejecutar la función
ampliar_dibujo()

C++
#include <iostream>
#include <vector>
#include <string>
#include <algorithm>
using namespace std;
int main() {
    vector<string> lineas(10);
    for (int i = 0; i < 10; i++) {
        getline(cin, lineas[i]);
    }
    cout << "-----" << endl;
    int filas = lineas.size();
    int columnas = 0;
    for (auto &linea : lineas) {
        columnas = max(columnas, (int)linea.size());
    }
    // Crear matriz original
    vector<vector<char>> matriz(filas, vector<char>(columnas, ' '));
    for (int i = 0; i < filas; i++) {
        for (int j = 0; j < (int)lineas[i].size(); j++) {
            matriz[i][j] = lineas[i][j];
        }
    }
}
```



<CODEWARS2026/>
LEÓN

```
    }  
}  
// Nueva matriz con borde adicional  
int nuevaFilas = filas;  
int nuevaColumnas = columnas + 1;  
vector<vector<char>> nuevaMatriz(nuevaFilas,  
vector<char>(nuevaColumnas, ' '));  
// Copiar las X  
for (int i = 0; i < filas; i++) {  
    for (int j = 0; j < columnas; j++) {  
        if (matriz[i][j] == 'X') {  
            nuevaMatriz[i][j] = 'X';  
        }  
    }  
}  
// Añadir 0 alrededor  
for (int i = 0; i < nuevaFilas; i++) {  
    for (int j = 0; j < nuevaColumnas; j++) {  
        if (nuevaMatriz[i][j] == 'X') {  
            for (int di = -1; di <= 1; di++) {  
                for (int dj = -1; dj <= 1; dj++) {  
                    int ni = i + di, nj = j + dj;  
                    if (ni >= 0 && ni < nuevaFilas && nj >= 0 && nj <  
nuevaColumnas) {  
                        if (nuevaMatriz[ni][nj] == ' ') {  
                            nuevaMatriz[ni][nj] = '0';  
                        }  
                    }  
                }  
            }  
        }  
    }  
}
```



<CODEWARS2026/>
LEÓN

```
        }
    }
}

// Imprimir resultado
for (int i = 0; i < nuevaFilas; i++) {
    string fila(nuevaMatriz[i].begin(), nuevaMatriz[i].end());
    // Eliminar espacios a la derecha
    fila.erase(find_if(fila.rbegin(), fila.rend(), [](unsigned char
ch) {
        return ch != ' ';
    }).base(), fila.end());
    cout << fila << endl;
}
return 0;
}
```

Java

```
import java.util.*;

public class Line_Amplifier {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String[] lineas = new String[10];
        // Leer 10 líneas
        for (int i = 0; i < 10; i++) {
            lineas[i] = sc.nextLine();
        }
        System.out.println("-----");
    }
}
```

<let's code/>





<CODEWARS2026/>
LEÓN

```
int filas = lineas.length;
int columnas = 0;
for (String linea : lineas) {
    columnas = Math.max(columnas, linea.length());
}
// Crear matriz original
char[][] matriz = new char[filas][columnas];
for (int i = 0; i < filas; i++) {
    String linea = lineas[i];
    for (int j = 0; j < columnas; j++) {
        matriz[i][j] = (j < linea.length()) ? linea.charAt(j) : '
';
    }
}
// Nueva matriz con borde adicional
int nuevaFilas = filas;
int nuevaColumnas = columnas + 1;
char[][] nuevaMatriz = new char[nuevaFilas][nuevaColumnas];
for (int i = 0; i < nuevaFilas; i++) {
    Arrays.fill(nuevaMatriz[i], ' ');
}
// Copiar las X
for (int i = 0; i < filas; i++) {
    for (int j = 0; j < columnas; j++) {
        if (matriz[i][j] == 'X') {
            nuevaMatriz[i][j] = 'X';
        }
    }
}
```




<CODEWARS2026/>
LEÓN

```
    }  
    // Añadir 0 alrededor  
    for (int i = 0; i < nuevaFilas; i++) {  
        for (int j = 0; j < nuevaColumnas; j++) {  
            if (nuevaMatriz[i][j] == 'X') {  
                for (int di = -1; di <= 1; di++) {  
                    for (int dj = -1; dj <= 1; dj++) {  
                        int ni = i + di, nj = j + dj;  
                        if (ni >= 0 && ni < nuevaFilas && nj >= 0 &&  
nj < nuevaColumnas) {  
                            if (nuevaMatriz[ni][nj] == ' ') {  
                                nuevaMatriz[ni][nj] = '0';  
                            }  
                        }  
                    }  
                }  
            }  
        }  
    }  
    // Imprimir resultado  
    for (int i = 0; i < nuevaFilas; i++) {  
        System.out.println(new  
String(nuevaMatriz[i]).stripTrailing());  
    }  
}
```

9

Tie Break

6 points

Introduction

You work as a programmer for an IT company, and the Roland Garros tennis tournament has commissioned your team to develop a project for an automatic scoreboard that doesn't require manual intervention.

Exercise

One person in your company is responsible for the visual design, another ensures that the AI detects who wins each point, another handles the point counting during the set, and you've been asked to create a program that counts the points during tie-breaks.



As is well known, the scoring system in tennis matches is peculiar: players score 15, 30, 40 points... But in tie-breaks, the counting is normal: 1, 2, 3, 4... So you breathe a sigh of relief.

But then you discover that tie-breaks also have their own rules: The tie-break is won by the first player to reach 7 points with a 2-point lead. This means that if the score is 7-6, the match isn't over yet and will continue to 8. If the player with 6 wins the next point, the score becomes 7-7 and will extend to 9... In a tie-break, the score could easily end up being 19-17 without any problem.

Input

As it's well-known computers only work with 0's and 1's. The AI will therefore only return a list of numbers (0 or 1) representing which player won each point (0 means that player 1 won the point, while 1 means that player 2 won the point). An example input might be:

```
0,1,0,0,1,1,1,0,0,1,0,0
```

Output

The program will generate an output in the following format (in spanish):

```
Jugador <GANADOR> gana el tie-break <RESULTADO>
```

For example, for the given input, the output would be:

```
Jugador 1 gana el tie-break 7-5
```

It could happen that the tie-break has not finished yet. In that case, the output will look like this:

```
El tie-break aún no ha terminado. Van 4-3
```

In this case, the first number corresponds to Player 1's points and the second to Player 2's points.

Example1

Input

```
0,1,0,0,1,1,1,0,0,1,0,0
```

Output

Jugador 1 gana el tie-break 7-5

Example2

Input

0,1,0,0,1,1,1,0,0,1,0,1,0,1,0,0,1,1

Output

Jugador 1 gana el tie-break 10-8

Solutions

Python

```
def tie_break_result(puntos):
    jugador1 = jugador2 = 0
    for punto in puntos:
        if punto == 0:
            jugador1 += 1
        elif punto == 1:
            jugador2 += 1
    # Comprobar si alguien ha ganado el tie-break
    if jugador1 >= 7 and jugador1 - jugador2 >= 2:
        return f"Jugador 1 gana el tie-break {jugador1}-{jugador2}"
    if jugador2 >= 7 and jugador2 - jugador1 >= 2:
        return f"Jugador 2 gana el tie-break {jugador2}-{jugador1}"
    # Si nadie ha ganado aún
    return f"El tie-break aún no ha terminado. Van {jugador1}-{jugador2}"

entrada = input("")
puntos = [int(x) for x in entrada.split(",")]
```




<CODEWARS2026/>
LEÓN

```
print(tie_break_result(puntos))
```

C++

```
#include <iostream>
#include <sstream>
#include <vector>
#include <string>
using namespace std;
string tieBreakResult(const vector<int>& puntos) {
    int jugador1 = 0, jugador2 = 0;
    for (int punto : puntos) {
        if (punto == 0) {
            jugador1++;
        } else if (punto == 1) {
            jugador2++;
        }
        // Comprobar si alguien ha ganado el tie-break
        if (jugador1 >= 7 && jugador1 - jugador2 >= 2) {
            return "Jugador 1 gana el tie-break " + to_string(jugador1) +
                "-" + to_string(jugador2);
        }
        if (jugador2 >= 7 && jugador2 - jugador1 >= 2) {
            return "Jugador 2 gana el tie-break " + to_string(jugador2) +
                "-" + to_string(jugador1);
        }
    }
    return "El tie-break aún no ha terminado. Van " + to_string(jugador1)
        + "-" + to_string(jugador2);
}
```

<let's code/>





<CODEWARS2026/>
LEÓN

```
int main() {
    string entrada;
    getline(cin, entrada);
    stringstream ss(entrada);
    string token;
    vector<int> puntos;
    while (getline(ss, token, ',')) {
        puntos.push_back(stoi(token));
    }
    cout << tieBreakResult(puntos) << endl;
    return 0;
}
```

Java

```
import java.util.Scanner;

public class Tiebreak {
    public static String tieBreakResult(int[] puntos) {
        int jugador1 = 0, jugador2 = 0;
        for (int punto : puntos) {
            if (punto == 0) {
                jugador1++;
            } else if (punto == 1) {
                jugador2++;
            }
        }
        // Comprobar si alguien ha ganado el tie-break
        if (jugador1 >= 7 && jugador1 - jugador2 >= 2) {
            return "Jugador 1 gana el tie-break " + jugador1 + "-" +
jugador2;
        }
    }
}
```

<let's code/>





<CODEWARS2026/>
LEÓN

```
        if (jugador2 >= 7 && jugador2 - jugador1 >= 2) {
            return "Jugador 2 gana el tie-break " + jugador2 + "-" +
jugador1;
        }
    }
    return "El tie-break aún no ha terminado. Van " + jugador1 + "-" +
jugador2;
}
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    String entrada = sc.nextLine();
    String[] partes = entrada.split(",");
    int[] puntos = new int[partes.length];
    for (int i = 0; i < partes.length; i++) {
        puntos[i] = Integer.parseInt(partes[i].trim());
    }
    System.out.println(tieBreakResult(puntos));
}
}
```

10

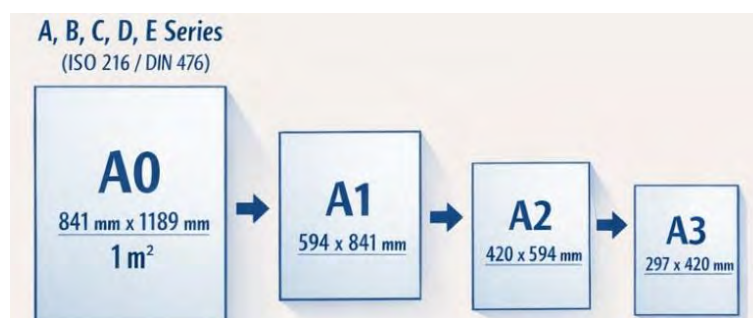
The ISO 216

7 points

Introduction

The printing paper format has standardized dimensions, which helps to unify production and printing criteria, and at the same time serves as a guide in design, graphic arts in design, graphic arts, advertising, press, etc. There are two main classifications of formats:

- European format, widely used in many countries around the world, includes the A, B, C, D and E series defined in ISO 216, which is based on the DIN 476 standard defined in 1922.
- American format, mainly used in the United States, Canada and Mexico, includes traditional formats such as letter, legal and tabloid.



The purpose behind the standardization of formats is to maximize paper usage to make the best possible use of it and minimize waste.

The ISO 216 standard defines paper formats whose sides maintain a constant ratio, such that when a sheet is divided in half by its length, each half still has the same ratio between its sides as the original sheet.

As a summary, for ISO/DIN formats:

- The ratio between their sides (x/y) is constant.
- The area and the sides lengths are expressed in square millimeters (mm^2) and millimeters (mm), respectively, and always rounded down to the nearest integer.

Exercise

You are asked to develop a program that, starting from the ISO A series whose base format is A0, which has a surface of 1 m^2 and dimensions of $841 \times 1189 \text{ mm}$, is able to calculate the approximate area of any successive format (A1, A2, A3, A4, A5, A6, A7 or A8).

These derived formats, whose number indicates the number of times the original sheet has been cut in half, have an area equal to half that of the previous format. Thus, an A1 sheet has an area of $\frac{1}{2} \text{ m}^2$, an A2 sheet has an area of $\frac{1}{4} \text{ m}^2$, and so on.

In addition to calculating the surface area of the specified format in mm^2 , the program must also provide the length of each side in millimeters (mm). In addition, It must also verify whether the input corresponds to a valid A format; otherwise, the following message must be displayed:

The entry does not correspond to a format A.

Note: All values should be rounded down to the nearest whole number.

Note 2: There is a simple relationship between the dimensions of successive formats. If a sheet of format A1 has dimensions $x \times y$ (with y being the shorter side), then a sheet of format A2 will have dimensions $y \times z$, where y remains the same value and becomes the longer side.

Input

A single line like A1, A2, A3, A4, A5...

Output

A single line with the following text:

Superficie: <area>; Lado mayor: <longer side>; Lado menor: <shorter side>

Example 1

Input

A1

Output

Superficie: 500000; Lado mayor: 841; Lado menor: 594

Example 2

Input

A5

Output

Superficie: 31250; Lado mayor: 210; Lado menor: 148

Solutions

Python

```
def calculate_format_area_and_dimensions(format_str: str):  
    # Validate input format  
    if not format_str.startswith('A') or not format_str[1:].isdigit():  
        print("The entry does not correspond to a format A.")  
    return
```



```
# Extract the number from the format (e.g., 'A1' -> 1)
n = int(format_str[1:])
# Ensure the number is between 0 and 8 (A0 to A8)
if n < 0 or n > 8:
    print("The entry does not correspond to a format A.")
    return
# Initial dimensions and area for A0
width = 841
height = 1189
area = 1000000 # in mm² (1 m² = 1000000 mm²)
# Calculate the format's area and dimensions based on A0
for i in range(n):
    # The shorter side of the previous sheet becomes the longer side
of the next
    # The longer side is halved to become the new shorter side
    height, width = width, height // 2
    area = area // 2 # Halve the area
# Output the results
print(f"Superficie: {area}; Lado mayor: {height}; Lado menor:
{width}")
# Main function to get the input and call the calculation function
def main():
    input_format = input().strip() # Read the input
    calculate_format_area_and_dimensions(input_format)
# Run the program
if __name__ == "__main__":
    main()
```

C++



<CODEWARS2026/> LEÓN

```
#include <iostream>
#include <string>
#include <cstdlib> // For std::stoi
using namespace std;
// Function to calculate format area and dimensions
void calculateFormatAreaAndDimensions(const string& format_str) {
    // Validate input format
    if (format_str.empty() || format_str[0] != 'A' ||
format_str.substr(1).find_first_not_of("0123456789") != string::npos) {
        cout << "The entry does not correspond to a format A." << endl;
        return;
    }
    // Extract the number from the format (e.g., 'A1' -> 1)
    int n;
    try {
        n = stoi(format_str.substr(1)); // Convert the substring to
integer
    } catch (const invalid_argument&) {
        cout << "The entry does not correspond to a format A." << endl;
        return;
    }
    // Ensure the number is between 0 and 8 (A0 to A8)
    if (n < 0 || n > 8) {
        cout << "The entry does not correspond to a format A." << endl;
        return;
    }
    // Initial dimensions and area for A0
    int width = 841;
```




<CODEWARS2026/>
LEÓN

```
int height = 1189;
int area = 1000000; // in mm² (1 m² = 1000000 mm²)
// Calculate the format's area and dimensions based on A0
for (int i = 0; i < n; ++i) {
    // For each step, we swap dimensions and halve the larger side
    if (width > height) {
        swap(width, height);
    }
    height = height / 2; // Halve the height
    area = area / 2; // Halve the area
}
// Output the results
cout << "Superficie: " << area << "; Lado mayor: " << max(width,
height) << "; Lado menor: " << min(width, height) << endl;
}
int main() {
    string input_format;
    getline(cin, input_format); // Read the input
    calculateFormatAreaAndDimensions(input_format); // Call the
calculation function
    return 0;
}
```

Java

```
import java.util.Scanner;
public class FormatDimensions {
    // Function to calculate format area and dimensions
    public static void calculateFormatAreaAndDimensions(String formatStr)
{
```

```
// Validate input format
if (formatStr.isEmpty() || formatStr.charAt(0) != 'A' ||
!formatStr.substring(1).matches("\\d+")) {
    System.out.println("The entry does not correspond to a format
A.");
    return;
}
// Extract the number from the format (e.g., 'A1' -> 1)
int n;
try {
    n = Integer.parseInt(formatStr.substring(1)); // Convert the
substring to integer
} catch (NumberFormatException e) {
    System.out.println("The entry does not correspond to a format
A.");
    return;
}
// Ensure the number is between 0 and 8 (A0 to A8)
if (n < 0 || n > 8) {
    System.out.println("The entry does not correspond to a format
A.");
    return;
}
// Initial dimensions and area for A0
int width = 841;
int height = 1189;
int area = 1000000; // in mm² (1 m² = 1000000 mm²)
// Calculate the format's area and dimensions based on A0
```



<CODEWARS2026/>
LEÓN

```
for (int i = 0; i < n; i++) {  
    // For each step, we swap dimensions and halve the larger side  
    if (width > height) {  
        int temp = width;  
        width = height;  
        height = temp;  
    }  
    height = height / 2; // Halve the height  
    area = area / 2; // Halve the area  
}  
// Output the results  
System.out.printf("Superficie: %d; Lado mayor: %d; Lado menor:  
%d\n", area, Math.max(width, height), Math.min(width, height));  
}  
public static void main(String[] args) {  
    Scanner scanner = new Scanner(System.in);  
    // Read the input  
    String inputFormat = scanner.nextLine().trim(); // Read the input  
    format and trim any extra spaces  
    // Call the calculation function  
    calculateFormatAreaAndDimensions(inputFormat);  
}  
}
```

<let's code/>



11

Rock, Paper, Scissors

7 points

Introduction

Son Goku and Vegeta, two of the most powerful Saiyan warriors, are preparing for the ultimate battle against Perfect Cell in his self-declared Martial Arts Tournament. However, there's only one spot for the next challenger, and Son Goku and Vegeta can't decide who should fight first.



To settle this, they decide to play a best of three "Rock, Paper, Scissors" game. You are Son Goku and the computer will be Vegeta. You must compete against the other computer in this decisive game. The battle continues until you or the computer wins at least 2 games (Best of 3 possible winning results are: (1-2), (2-1), (2-0), (0-2)).

Exercise

Between you and me, Vegeta is quite foolish: it will always reply in a specific order. In the first turn, it will always show "Scissors", in the second turn it will show "Rock" and in the third turn it will always show "Paper", and then in the fourth turn it will show again "Scissors", and in the fifth it will show "Rock" again, and so on.

You are asked to write a program that read the Son Goku selections and play against Vegeta (computer) until one of the wins (or the game end in a draw). The read selections will play against the computer's selections that will always be the same: SRPSRPSRPSRP...

The program will simply write the name of the Winner or if it's been a draw (See explanations below).

Input

The program will read a single line with letters P, R or S, that specific the decisions (Paper,Rock,Scissors) that Son Goku would take in possible future. For instance:

```
RPSRPSRPSRPSRPSRPSR
```

Output

The program will simply write the name of the Winner:

```
SON GOKU WILL FIGHT
```

or

```
VEGETA WILL FIGHT
```

or

```
IT'S A DRAW; PLAY AGAIN
```

Example 1

Input

RPSRPSRPSRPSRPSRPSR

Output

SON GOKU WILL FIGHT

Example 2

Input

SRPSRPSRP

Output

IT'S A DRAW; PLAY AGAIN

Example 3

Input

SRPPSP

Output

VEGETA WILL FIGHT

Solutions

Python

```
def determine_winner(son_goku_move, vegeta_move):  
    # Determine the winner of one round  
    if son_goku_move == vegeta_move:  
        return 'draw' # It's a draw if both moves are the same  
    elif (son_goku_move == 'R' and vegeta_move == 'S') or \  
        (son_goku_move == 'S' and vegeta_move == 'P') or \  
        (son_goku_move == 'P' and vegeta_move == 'R'):
```



```
        return 'son_goku' # Son Goku wins
    else:
        return 'vegeta' # Vegeta wins
def main():
    # Read the input string (Son Goku's sequence of moves)
    son_goku_moves = input().strip()
    # Vegeta's moves cycle: Scissors -> Rock -> Paper -> Scissors -> Rock
    -> Paper ...
    vegeta_moves = ['S', 'R', 'P']
    # Initialize counters for wins
    son_goku_wins = 0
    vegeta_wins = 0
    round_count = 0
    # We only need to play until one player wins 2 rounds
    for i in range(len(son_goku_moves)):
        son_goku_move = son_goku_moves[i]
        vegeta_move = vegeta_moves[i % 3] # Cycle through Vegeta's moves
        #print(son_goku_move, " VS ", vegeta_move)
        # Determine the winner of this round
        result = determine_winner(son_goku_move, vegeta_move)
        if result == 'son_goku':
            son_goku_wins += 1
        elif result == 'vegeta':
            vegeta_wins += 1
        #print(son_goku_wins, " VS ", vegeta_wins)
        # Check if any player has won 2 rounds
        if son_goku_wins == 2:
            print("SON GOKU WILL FIGHT")
```



<CODEWARS2026/>
LEÓN

```
        return
    elif vegeta_wins == 2:
        print("VEGETA WILL FIGHT")
        return
    round_count += 1
    # If we finish all rounds without a 2-0 decision, it's a draw
    print("IT'S A DRAW; PLAY AGAIN")
# Execute the main function
main()
```

C++

```
#include <iostream>
#include <string>
using namespace std;
string determine_winner(char son_goku_move, char vegeta_move) {
    // Determine the winner of one round
    if (son_goku_move == vegeta_move) {
        return "draw"; // It's a draw if both moves are the same
    } else if ((son_goku_move == 'R' && vegeta_move == 'S') ||
               (son_goku_move == 'S' && vegeta_move == 'P') ||
               (son_goku_move == 'P' && vegeta_move == 'R')) {
        return "son_goku"; // Son Goku wins
    } else {
        return "vegeta"; // Vegeta wins
    }
}
int main() {
    // Read the input string (Son Goku's sequence of moves)
    string son_goku_moves;
```




<CODEWARS2026/>
LEÓN

```
cin >> son_goku_moves;
// Vegeta's moves cycle: Scissors -> Rock -> Paper -> Scissors -> Rock
-> Paper ...
char vegeta_moves[] = {'S', 'R', 'P'};
int son_goku_wins = 0;
int vegeta_wins = 0;
// Play the game until one player wins 2 rounds
for (int i = 0; i < son_goku_moves.length(); i++) {
    char son_goku_move = son_goku_moves[i];
    char vegeta_move = vegeta_moves[i % 3]; // Cycle through Vegeta's
moves
    // Determine the winner of this round
    string result = determine_winner(son_goku_move, vegeta_move);
    if (result == "son_goku") {
        son_goku_wins++;
    } else if (result == "vegeta") {
        vegeta_wins++;
    }
    // Check if any player has won 2 rounds
    if (son_goku_wins == 2) {
        cout << "SON GOKU WILL FIGHT" << endl;
        return 0;
    } else if (vegeta_wins == 2) {
        cout << "VEGETA WILL FIGHT" << endl;
        return 0;
    }
}
// If we finish all rounds without a 2-0 decision, it's a draw
```



```
    cout << "IT'S A DRAW; PLAY AGAIN" << endl;  
    return 0;  
}
```

Java

```
import java.util.Scanner;  
  
public class RockPaperScissors {  
    // Determine the winner of one round  
    public static String determineWinner(char sonGokuMove, char  
vegetaMove) {  
        if (sonGokuMove == vegetaMove) {  
            return "draw"; // It's a draw if both moves are the same  
        } else if ((sonGokuMove == 'R' && vegetaMove == 'S') ||  
                    (sonGokuMove == 'S' && vegetaMove == 'P') ||  
                    (sonGokuMove == 'P' && vegetaMove == 'R')) {  
            return "son_goku"; // Son Goku wins  
        } else {  
            return "vegeta"; // Vegeta wins  
        }  
    }  
  
    public static void main(String[] args) {  
        // Read the input string (Son Goku's sequence of moves)  
        Scanner sc = new Scanner(System.in);  
        String sonGokuMoves = sc.nextLine();  
        sc.close();  
        // Vegeta's moves cycle: Scissors -> Rock -> Paper -> Scissors ->  
Rock -> Paper ...  
        char[] vegetaMoves = {'S', 'R', 'P'};  
        int sonGokuWins = 0;
```

```
int vegetaWins = 0;
// Play the game until one player wins 2 rounds
for (int i = 0; i < sonGokuMoves.length(); i++) {
    char sonGokuMove = sonGokuMoves.charAt(i);
    char vegetaMove = vegetaMoves[i % 3]; // Cycle through
Vegeta's moves
    // Determine the winner of this round
    String result = determineWinner(sonGokuMove, vegetaMove);
    if (result.equals("son_goku")) {
        sonGokuWins++;
    } else if (result.equals("vegeta")) {
        vegetaWins++;
    }
    // Check if any player has won 2 rounds
    if (sonGokuWins == 2) {
        System.out.println("SON GOKU WILL FIGHT");
        return;
    } else if (vegetaWins == 2) {
        System.out.println("VEGETA WILL FIGHT");
        return;
    }
}
// If we finish all rounds without a 2-0 decision, it's a draw
System.out.println("IT'S A DRAW; PLAY AGAIN");
}
}
```


12

The time-traveling gunslinger

8 points

Introduction

In a distant past, a gunslinger named Rex, the Time Traveler, has been assigned a very special mission. He is known not only for his skill with weapons, but also for his ability to travel through time and space. His mission is crucial: to ensure that historical events happen as they should, without alterations.



To complete his task, Rex must travel to different eras and places around the world, and he often encounters an unexpected difficulty: date formats vary depending on the time period and the location.

For example, in the future, dates are written in the format YYYY-MM-DD, while in the past, some cultures write dates in the format DD/MM/YYYY, and in ancient civilizations, dates may be represented in entirely different ways.

Rex needs your help to easily convert dates from one format to another. You will be his trusted programmer, responsible for writing the algorithm that allows him to switch between date formats depending on the era he is in.

Exercise

Your task is to create a program that receives a date in a specific format and converts it to another format that Rex needs to complete his mission. The program must read the date in the input format, validate it, and then convert it to the output format.

The date formats Rex needs to handle are:

```
"YYYY-MM-DD" (e.g., "2024-12-10")  
"DD/MM/YYYY" (e.g., "10/12/2024")
```

Input

The program will receive a single line of text in the format:

```
source_date_format --> destination_date_format --> date, where:
```

where:

- source_date_format can be "YYYY-MM-DD" or "DD/MM/YYYY".
- destination_date_format will also be one of these two formats.
- date is the date to be converted.

```
YYYY-MM-DD --> DD/MM/YYYY --> 2014-05-21
```

Before the first arrow, you can see the 'original' format. Then the destination format and finally the date we want to convert (Hopefully in the 'original' format).

Output

The program should be able to convert a date from "YYYY-MM-DD" to "DD/MM/YYYY" and vice versa.

If the date format is invalid, or if the values for the day, month, or year are incorrect (e.g., if the day is 32 or the month is 13), the program should return the message "Invalid date".

Example 1

Input

```
YYYY-MM-DD --> DD/MM/YYYY --> 2014-05-21
```

Output

```
21/05/2014
```

Example 2

Input

```
DD/MM/YYYY --> YYYY-MM-DD --> 21/05/2014
```

Output

```
2014-05-21
```

Example 3

Input

```
DD/MM/YYYY --> DD/MM/YYYY --> 20/20/2020
```

Output

```
Invalid date
```

Solutions

Python

```
from datetime import datetime

def convert_date(input_line: str):
    # Split the input by the arrow symbols
    parts = input_line.strip().split("-->")
```



<CODEWARS2026/>
LEÓN

```
if len(parts) != 3:
    return "Invalid date"
source_format = parts[0].strip()
destination_format = parts[1].strip()
date_str = parts[2].strip()
# Map the readable format to Python's datetime format
format_map = {
    "YYYY-MM-DD": "%Y-%m-%d",
    "DD/MM/YYYY": "%d/%m/%Y"
}
# Validate format presence
if source_format not in format_map or destination_format not in
format_map:
    return "Invalid date"
# Try to parse the input date according to the source format
try:
    date_obj = datetime.strptime(date_str,
format_map[source_format])
    # Convert to the destination format
    converted_date =
date_obj.strftime(format_map[destination_format])
    return converted_date
except ValueError:
    return "Invalid date"
# Example usage:
if __name__ == "__main__":
    # You can test with inputs like:
    # "YYYY-MM-DD --> DD/MM/YYYY --> 2024-12-10"
```

<let's code/>





<CODEWARS2026/> LEÓN

```
# "DD/MM/YYYY --> YYYY-MM-DD --> 10/12/2024"  
user_input = input().strip()  
print(convert_date(user_input))
```

C++

```
#include <iostream>  
#include <sstream>  
#include <string>  
#include <vector>  
#include <iomanip>  
#include <ctime>  
using namespace std;  
  
// Función auxiliar para dividir un string por un delimitador  
vector<string> split(const string &str, const string &delimiter) {  
    vector<string> tokens;  
    size_t start = 0, end = 0;  
    while ((end = str.find(delimiter, start)) != string::npos) {  
        tokens.push_back(str.substr(start, end - start));  
        start = end + delimiter.length();  
    }  
    tokens.push_back(str.substr(start));  
    return tokens;  
}  
  
// Verifica si una fecha es válida  
bool isValidDate(int day, int month, int year) {  
    if (year < 1 || month < 1 || month > 12 || day < 1) return false;  
    static const int daysInMonth[] = { 31,28,31,30,31,30,31,31,30,31,30,31  
};  
    int days = daysInMonth[month - 1];
```




<CODEWARS2026/> LEÓN

```
// Año bisiesto
if (month == 2) {
    bool leap = (year % 4 == 0 && (year % 100 != 0 || year % 400 ==
0));
    if (leap) days = 29;
}
return day <= days;
}

string convertDate(const string &inputLine) {
    vector<string> parts = split(inputLine, "-->");
    if (parts.size() != 3) return "Invalid date";
    string sourceFormat = parts[0];
    string destFormat = parts[1];
    string dateStr = parts[2];
    // Eliminar espacios
    auto trim = [](string &s) {
        s.erase(0, s.find_first_not_of(" \t\n\r"));
        s.erase(s.find_last_not_of(" \t\n\r") + 1);
    };
    trim(sourceFormat);
    trim(destFormat);
    trim(dateStr);
    int day, month, year;
    // Leer según el formato de origen
    if (sourceFormat == "YYYY-MM-DD") {
        if (sscanf(dateStr.c_str(), "%d-%d-%d", &year, &month, &day) != 3)
            return "Invalid date";
    } else if (sourceFormat == "DD/MM/YYYY") {
```



```
        if (sscanf(dateStr.c_str(), "%d/%d/%d", &day, &month, &year) != 3)
            return "Invalid date";
    } else {
        return "Invalid date";
    }
    // Validar la fecha
    if (!isValidDate(day, month, year)) return "Invalid date";
    // Convertir según el formato de destino
    ostringstream output;
    if (destFormat == "YYYY-MM-DD") {
        output << setw(4) << setfill('0') << year << "-"
            << setw(2) << setfill('0') << month << "-"
            << setw(2) << setfill('0') << day;
    } else if (destFormat == "DD/MM/YYYY") {
        output << setw(2) << setfill('0') << day << "/"
            << setw(2) << setfill('0') << month << "/"
            << setw(4) << setfill('0') << year;
    } else {
        return "Invalid date";
    }
    return output.str();
}

int main() {
    string inputLine;
    getline(cin, inputLine);
    cout << convertDate(inputLine) << endl;
    return 0;
}
```

Java

```
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.time.format.DateTimeParseException;
import java.util.Scanner;

public class DateConverter {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String input = scanner.nextLine().trim();
        scanner.close();
        System.out.println(convertDate(input));
    }
    public static String convertDate(String inputLine) {
        try {
            String[] parts = inputLine.split("-->");
            if (parts.length != 3) {
                return "Invalid date";
            }
            String sourceFormat = parts[0].trim();
            String destinationFormat = parts[1].trim();
            String dateStr = parts[2].trim();
            DateTimeFormatter sourceFormatter;
            DateTimeFormatter destFormatter;
            switch (sourceFormat) {
                case "YYYY-MM-DD":
                    sourceFormatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");
                    break;
```



<CODEWARS2026/>
LEÓN

```
        case "DD/MM/YYYY":
            sourceFormatter =
DateTimeFormatter.ofPattern("dd/MM/yyyy");
            break;
        default:
            return "Invalid date";
    }
    switch (destinationFormat) {
        case "YYYY-MM-DD":
            destFormatter = DateTimeFormatter.ofPattern("yyyy-MM-
dd");

            break;
        case "DD/MM/YYYY":
            destFormatter =
DateTimeFormatter.ofPattern("dd/MM/yyyy");
            break;
        default:
            return "Invalid date";
    }
    LocalDate date;
    try {
        date = LocalDate.parse(dateStr, sourceFormatter);
    } catch (DateTimeParseException e) {
        return "Invalid date";
    }
    return date.format(destFormatter);
} catch (Exception e) {
    return "Invalid date";
}
```

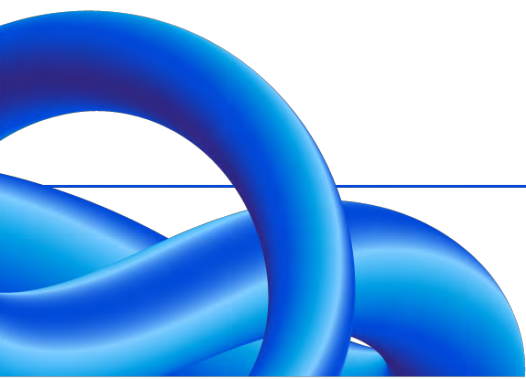
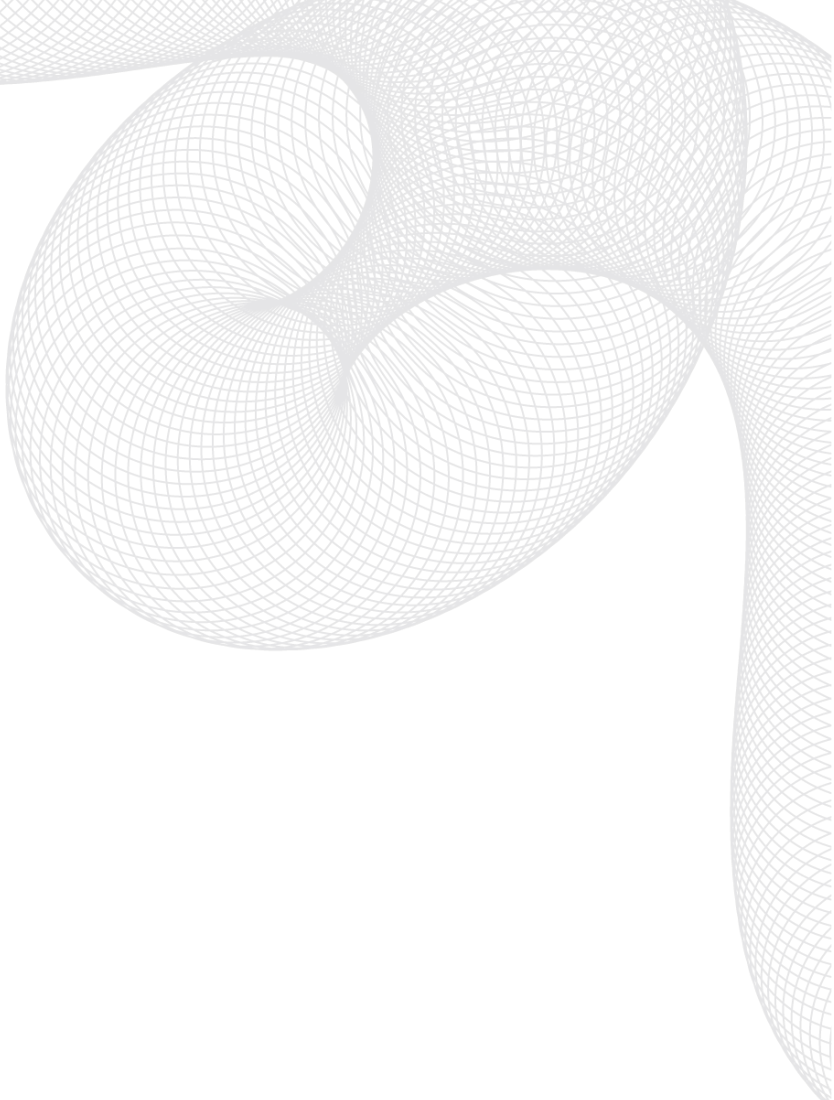
<let's code/>





<CODEWARS2026/>
LEÓN

```
}  
}  
}
```



<let's code/>



13

Yo soy tu primo

8 points

Introduction

In the Cloud City, Luke Skywalker faces Darth Vader in a decisive duel. In the midst of the battle, Darth Vader is about to reveal a shocking secret: they could be cousins!

But it's not a matter of blood relation; it's a mathematical one. Your mission is to write a program that helps Luke discover whether Darth Vader is telling the truth.



Exercise

The program should do the following:

- > Receive a number that represents the Force power of Luke or Darth Vader.
- > Determine if the entered number is prime.

-> If the number is prime, display the message: "It's true! This number is prime. The Force connection is real."

-> If the number is not prime, display: "The number is not prime. Darth Vader is lying."

-> If the number is 0, the program should finish with the message: "May the Force be with you, young padawan."

-> Otherwise, write "The entered number is incorrect, young padawan."

Input

The program will read a single text line.

Output

The output will be a single text line following the instructions given earlier.

Example 1

Input

7

Output

It's true! The number 7 is prime. The connection to the Force is real.

Example 2

Input

10

Output

The number 10 is not prime. Darth Vader lies.

Example 3

Input

0

Output

May the Force be with you, young padawan.

Example 4

Input

abc

Output

The entered number is incorrect, young padawan.

Solutions

Python

```
def is_prime(number):  
    if number <= 1:  
        return False  
    for i in range(2, int(number ** 0.5) + 1):  
        if number % i == 0:  
            return False  
    return True  
  
try:  
    number = int(input())  
    if number == 0:
```



```
        print("May the Force be with you, young padawan.")
    elif is_prime(number):
        print(f"It's true! The number {number} is prime. The
connection to the Force is real.")
    else:
        print(f"The number {number} is not prime. Darth Vader lies.")
except ValueError:
    print("The entered number is incorrect, young padawan.")
```

C++

```
#include <iostream>
#include <cmath>
using namespace std;
bool isPrime(long long number) {
    if (number <= 1) return false;
    for (long long i = 2; i <= sqrt(number); ++i) {
        if (number % i == 0) return false;
    }
    return true;
}
int main() {
    long long number;
    if (!(cin >> number)) {
        cout << "The entered number is incorrect, young padawan." <<
endl;

        cin.clear();
        cin.ignore(10000, '\n');
    }
    else if (number == 0) {
```



<CODEWARS2026/>
LEÓN

```
        cout << "May the Force be with you, young padawan." << endl;
    }
    else if (isPrime(number)) {
        cout << "It's true! The number " << number << " is prime. The
connection to the Force is real." << endl;
    } else {
        cout << "The number " << number << " is not prime. Darth Vader
lies." << endl;
    }
    return 0;
}
```

Java

```
import java.util.InputMismatchException;
import java.util.Scanner;
public class Yo_Soy_Tu_Primo
{
    /**
     * Check if a number is prime
     * @param number number to check
     * @return boolean if the number is prime or not
     */
    public static boolean isPrime(long number) {
        if (number <= 1) {
            return false;
        }
        for (int i = 2; i <= Math.sqrt(number); i++) {
            if (number % i == 0) {
                return false;
            }
        }
    }
}
```



<CODEWARS2026/>
LEÓN

```
        }
    }
    return true;
}

public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    try{
        long number;
        number = scanner.nextLong();
        if (number == 0) {
            System.out.println("May the Force be with you, young
padawan.");
        }
        else if (isPrime(number)) {
            System.out.println("It's true! The number " + number +
" is prime. The connection to the Force is real.");
        } else {
            System.out.println("The number " + number + " is not
prime. Darth Vader lies.");
        }
    } catch (InputMismatchException e) {
        System.out.println("The entered number is incorrect, young
padawan.");
        scanner.nextLine();
    }
}
}
```

<let's code/>



14

Formula 1 Tactical Manager

9 points

Introduction

You are an F1 expert strategy manager that needs to calculate the optimal tire strategy for a Formula 1 race. In Formula 1, there are three types of tires available, each with different characteristics:

- Soft tires: Faster lap times but wear out quickly
- Medium tires: Balanced performance and durability
- Hard tires: Slower lap times but last longer

Each tire type has two key parameters:

1. Lap time (in seconds): The time it takes to complete one lap with these tires
2. Maximum laps: The maximum number of laps these tires can last before they must be changed. There is no races with more than 30 laps.



Race Rules

1. Mandatory pit stop: At least one pit stop must be made during the race

2. Pit stop time: Each pit stop takes a fixed time of 20 seconds
3. Tire variety rule: At least TWO different tire types must be used during the race
4. Tire reuse: The same tire type can be used multiple times (in different stints)

Exercise

Write a program to find the tire strategy that minimizes the total race time (lap times + pit stop penalties).

Input

The program receives input from the keyboard (stdin) in the following format:

```
<soft_laptime>,<soft_maxlaps>-<medium_laptime>,<medium_maxlaps>-  
<hard_laptime>,<hard_maxlaps>  
<total_laps>
```

Where:

- Tire parameters are separated by commas , within each type
- Different tire types are separated by hyphens -
- The format is: *soft-medium-hard*
- Second line contains the total number of laps in the race

For instance:

```
120.13,10-130.26,20-140.53,30  
30
```

This means:

- Soft tires: 120.13 seconds per lap, maximum 10 laps
- Medium tires: 130.26 seconds per lap, maximum 20 laps
- Hard tires: 140.53 seconds per lap, maximum 30 laps

- Race distance: 30 laps

Output

The program should output to the screen (stdout):

```
<tire1>(<laps1>),<tire2>(<laps2>), ... <tireN>(<lapsN>)
```

Where:

- <tire> is one of: S (for Soft), M (for Medium), or H (For Hard)
- <laps> is the number of laps on that tire compound

With the previous input, the output would be:

```
S(9),M(1),S(10),S(10)
```

Example 1

Input

```
120.13,10-130.26,20-140.53,30  
30
```

Output

```
S(9),M(1),S(10),S(10)
```

Example 2

Input

```
90.5,5-95.2,10-100.8,15  
12
```

Output

```
S(5),S(5),M(2)
```

Solutions

Python

```
import sys
from typing import List, Tuple
class Tire:
    def __init__(self, name: str, lap_time: float = 0.0, max_laps: int =
0):
        self.name = name
        self.lap_time = lap_time
        self.max_laps = max_laps
class Strategy:
    def __init__(self):
        self.stints: List[Tuple[int, int]] = [] # List of
(tire_type_index, number_of_laps)
        self.total_time: float = float('inf')
PIT_STOP_TIME = 20.0
def parse_input(input_str: str) -> List[Tire]:
    """Parse input string to extract tire parameters"""
    tires = [
        Tire("S"),
        Tire("M"),
        Tire("H")
    ]
    tokens = input_str.split('-')
    for idx, token in enumerate(tokens[:3]):
        parts = token.split(',')
        tires[idx].lap_time = float(parts[0])
```



<CODEWARS2026/>
LEÓN

```
        tires[idx].max_laps = int(parts[1])
    return tires

def uses_at_least_two_types(stints: List[Tuple[int, int]]) -> bool:
    """Check if a strategy uses at least two different tire types"""
    if not stints:
        return False
    first_type = stints[0][0]
    for stint in stints:
        if stint[0] != first_type:
            return True
    return False

def calculate_time(tires: List[Tire], stints: List[Tuple[int, int]]) ->
float:
    """Calculate total time for a given strategy"""
    total_time = 0.0
    for tire_type, laps in stints:
        total_time += tires[tire_type].lap_time * laps
    total_time += (len(stints) - 1) * PIT_STOP_TIME
    return total_time

def generate_strategies(tires: List[Tire], total_laps: int, current_lap:
int,
                        current_stints: List[Tuple[int, int]],
best_strategy: Strategy) -> None:
    """Recursive function to generate all valid strategies"""
    # Base case: we've covered all laps
    if current_lap == total_laps:
        # Check if strategy is valid (at least 2 tire types and at least 1
        pit stop)
```



```
        if len(current_stints) >= 2 and
uses_at_least_two_types(current_stints):
            time = calculate_time(tires, current_stints)
            if time < best_strategy.total_time:
                best_strategy.stints = current_stints.copy()
                best_strategy.total_time = time
    return

# Try each tire type
for tire_type in range(3):
    # Try different stint lengths (1 to min(maxLaps, remaining laps))
    remaining_laps = total_laps - current_lap
    max_stint_laps = min(tires[tire_type].max_laps, remaining_laps)
    for stint_laps in range(1, max_stint_laps + 1):
        # Pruning: if we've already used too much time, skip
        current_stints.append((tire_type, stint_laps))
        # Only recurse if this could potentially be better
        current_time = calculate_time(tires, current_stints)
        remaining_after_stint = total_laps - current_lap - stint_laps
        # Simple lower bound: remaining laps with fastest tire
        min_possible_time = current_time
        if remaining_after_stint > 0:
            fastest_lap_time = min(tire.lap_time for tire in tires)
            min_possible_time += remaining_after_stint *
fastest_lap_time + PIT_STOP_TIME
        if min_possible_time < best_strategy.total_time:
            generate_strategies(tires, total_laps, current_lap +
stint_laps,

                                current_stints, best_strategy)
```



<CODEWARS2026/>
LEÓN

```
        current_stints.pop()

def main():
    # Read input
    tire_input = input().strip()
    total_laps = int(input().strip())
    # Parse tire data
    tires = parse_input(tire_input)
    # Find best strategy
    best_strategy = Strategy()
    current_stints: List[Tuple[int, int]] = []
    generate_strategies(tires, total_laps, 0, current_stints,
best_strategy)
    # Output result
    result = []
    for tire_type, laps in best_strategy.stints:
        result.append(f"{tires[tire_type].name}({laps})")
    print(",".join(result))
if __name__ == "__main__":
    main()
```

C++

```
#include <iostream>
#include <string>
#include <vector>
#include <sstream>
#include <limits>
#include <algorithm>
using namespace std;
struct Tire {
```

<let's code/>





<CODEWARS2026/> LEÓN

```
    string name;
    double lapTime;
    int maxLaps;
};
struct Strategy {
    vector<pair<int, int>> stints; // pair of (tire_type_index,
number_of_laps)
    double totalTime;
    Strategy() : totalTime(numeric_limits<double>::max()) {}
};
const double PIT_STOP_TIME = 20.0;
// Parse input string to extract tire parameters
vector<Tire> parseInput(const string& input) {
    vector<Tire> tires(3);
    tires[0].name = "S";
    tires[1].name = "M";
    tires[2].name = "H";
    stringstream ss(input);
    string token;
    int idx = 0;
    while (getline(ss, token, '-') && idx < 3) {
        size_t commaPos = token.find(',');
        tires[idx].lapTime = stod(token.substr(0, commaPos));
        tires[idx].maxLaps = stoi(token.substr(commaPos + 1));
        idx++;
    }
    return tires;
}
```



```
// Check if a strategy uses at least two different tire types
bool usesAtLeastTwoTypes(const vector<pair<int, int>>& stints) {
    if (stints.empty()) return false;
    int firstType = stints[0].first;
    for (const auto& stint : stints) {
        if (stint.first != firstType) {
            return true;
        }
    }
    return false;
}

// Calculate total time for a given strategy
double calculateTime(const vector<Tire>& tires, const vector<pair<int,
int>>& stints) {
    double totalTime = 0.0;
    for (const auto& stint : stints) {
        int tireType = stint.first;
        int laps = stint.second;
        totalTime += tires[tireType].lapTime * laps;
    }
    totalTime += (stints.size() - 1) * PIT_STOP_TIME;
    return totalTime;
}

// Recursive function to generate all valid strategies
void generateStrategies(const vector<Tire>& tires, int totalLaps, int
currentLap,
                        vector<pair<int, int>>& currentStints, Strategy&
bestStrategy) {
```




```
// Base case: we've covered all laps
if (currentLap == totalLaps) {
    // Check if strategy is valid (at least 2 tire types and at least
1 pit stop)
    if (currentStints.size() >= 2 &&
usesAtLeastTwoTypes(currentStints)) {
        double time = calculateTime(tires, currentStints);
        if (time < bestStrategy.totalTime) {
            bestStrategy.stints = currentStints;
            bestStrategy.totalTime = time;
        }
    }
    return;
}

// Try each tire type
for (int tireType = 0; tireType < 3; tireType++) {
    // Try different stint lengths (1 to min(maxLaps, remaining laps))
    int remainingLaps = totalLaps - currentLap;
    int maxStintLaps = min(tires[tireType].maxLaps, remainingLaps);
    for (int stintLaps = 1; stintLaps <= maxStintLaps; stintLaps++) {
        // Pruning: if we've already used too much time, skip
        currentStints.push_back({tireType, stintLaps});
        // Only recurse if this could potentially be better
        double currentTime = calculateTime(tires, currentStints);
        int remainingAfterStint = totalLaps - currentLap - stintLaps;
        // Simple lower bound: remaining laps with fastest tire
        double minPossibleTime = currentTime;
        if (remainingAfterStint > 0) {
```



<CODEWARS2026/>
LEÓN

```
        double fastestLapTime = min({tires[0].lapTime,
tires[1].lapTime, tires[2].lapTime});
        minPossibleTime += remainingAfterStint * fastestLapTime +
PIT_STOP_TIME;
    }
    if (minPossibleTime < bestStrategy.totalTime) {
        generateStrategies(tires, totallaps, currentLap +
stintLaps, currentStints, bestStrategy);
    }
    currentStints.pop_back();
}
}

int main() {
    string tireInput;
    int totallaps;
    // Read input
    getline(cin, tireInput);
    cin >> totallaps;
    // Parse tire data
    vector<Tire> tires = parseInput(tireInput);
    // Find best strategy
    Strategy bestStrategy;
    vector<pair<int, int>> currentStints;
    generateStrategies(tires, totallaps, 0, currentStints, bestStrategy);
    // Output result
    for (size_t i = 0; i < bestStrategy.stints.size(); i++) {
        int tireType = bestStrategy.stints[i].first;
```



<CODEWARS2026/>
LEÓN

```
int laps = bestStrategy.stints[i].second;
cout << tires[tireType].name << "(" << laps << ")";
if (i < bestStrategy.stints.size() - 1) {
    cout << ",";
}
}
cout << endl;
return 0;
}
```

Java

```
import java.util.*;
public class F1Strategy {
    static class Tire {
        String name;
        double lapTime;
        int maxLaps;
        Tire(String name, double lapTime, int maxLaps) {
            this.name = name;
            this.lapTime = lapTime;
            this.maxLaps = maxLaps;
        }
    }
    static class Stint {
        int tireTypeIndex;
        int numberOfLaps;
        Stint(int tireTypeIndex, int numberOfLaps) {
            this.tireTypeIndex = tireTypeIndex;
            this.numberOfLaps = numberOfLaps;
        }
    }
}
```



<CODEWARS2026/>
LEÓN

```
    }  
}  
static class Strategy {  
    List<Stint> stints;  
    double totalTime;  
    Strategy() {  
        this.stints = new ArrayList<>();  
        this.totalTime = Double.MAX_VALUE;  
    }  
}  
  
static final double PIT_STOP_TIME = 20.0;  
// Parse input string to extract tire parameters  
static List<Tire> parseInput(String input) {  
    List<Tire> tires = new ArrayList<>();  
    tires.add(new Tire("S", 0.0, 0));  
    tires.add(new Tire("M", 0.0, 0));  
    tires.add(new Tire("H", 0.0, 0));  
    String[] tokens = input.split("-");  
    for (int idx = 0; idx < Math.min(tokens.length, 3); idx++) {  
        String[] parts = tokens[idx].split(",");  
        tires.get(idx).lapTime = Double.parseDouble(parts[0]);  
        tires.get(idx).maxLaps = Integer.parseInt(parts[1]);  
    }  
    return tires;  
}  
  
// Check if a strategy uses at least two different tire types  
static boolean usesAtLeastTwoTypes(List<Stint> stints) {  
    if (stints.isEmpty()) return false;
```




<CODEWARS2026/>
LEÓN

```
int firstType = stints.get(0).tireTypeIndex;
for (Stint stint : stints) {
    if (stint.tireTypeIndex != firstType) {
        return true;
    }
}
return false;
}

// Calculate total time for a given strategy
static double calculateTime(List<Tire> tires, List<Stint> stints) {
    double totalTime = 0.0;
    for (Stint stint : stints) {
        int tireType = stint.tireTypeIndex;
        int laps = stint.numberOfLaps;
        totalTime += tires.get(tireType).lapTime * laps;
    }
    totalTime += (stints.size() - 1) * PIT_STOP_TIME;
    return totalTime;
}

// Recursive function to generate all valid strategies
static void generateStrategies(List<Tire> tires, int totalLaps, int
currentLap,

                                List<Stint> currentStints, Strategy
bestStrategy) {
    // Base case: we've covered all laps
    if (currentLap == totalLaps) {
        // Check if strategy is valid (at least 2 tire types and at
least 1 pit stop)
```

```
        if (currentStints.size() >= 2 &&
usesAtLeastTwoTypes(currentStints)) {
            double time = calculateTime(tires, currentStints);
            if (time < bestStrategy.totalTime) {
                bestStrategy.stints = new ArrayList<>(currentStints);
                bestStrategy.totalTime = time;
            }
        }
        return;
    }
    // Try each tire type
    for (int tireType = 0; tireType < 3; tireType++) {
        // Try different stint lengths (1 to min(maxLaps, remaining
laps))

        int remainingLaps = totallaps - currentLap;
        int maxStintLaps = Math.min(tires.get(tireType).maxLaps,
remainingLaps);
        for (int stintLaps = 1; stintLaps <= maxStintLaps;
stintLaps++) {
            // Pruning: if we've already used too much time, skip
currentStints.add(new Stint(tireType, stintLaps));
            // Only recurse if this could potentially be better
            double currentTime = calculateTime(tires, currentStints);
            int remainingAfterStint = totallaps - currentLap -
stintLaps;

            // Simple lower bound: remaining laps with fastest tire
            double minPossibleTime = currentTime;
            if (remainingAfterStint > 0) {
```

```
        double fastestLapTime =  
Math.min(Math.min(tires.get(0).lapTime,  
  
tires.get(1).lapTime),  
  
tires.get(2).lapTime);  
        minPossibleTime += remainingAfterStint *  
fastestLapTime + PIT_STOP_TIME;  
    }  
    if (minPossibleTime < bestStrategy.totalTime) {  
        generateStrategies(tires, totalLaps, currentLap +  
stintLaps,  
  
currentStints, bestStrategy);  
    }  
    currentStints.remove(currentStints.size() - 1);  
}  
}  
}  
  
public static void main(String[] args) {  
    Scanner scanner = new Scanner(System.in);  
    // Read input  
    String tireInput = scanner.nextLine();  
    int totalLaps = scanner.nextInt();  
    // Parse tire data  
    List<Tire> tires = parseInput(tireInput);  
    // Find best strategy  
    Strategy bestStrategy = new Strategy();  
    List<Stint> currentStints = new ArrayList<>();
```



<CODEWARS2026/>
LEÓN

```
        generateStrategies(tires, totallaps, 0, currentStints,
bestStrategy);
    // Output result
    for (int i = 0; i < bestStrategy.stints.size(); i++) {
        Stint stint = bestStrategy.stints.get(i);
        int tireType = stint.tireTypeIndex;
        int laps = stint.numberOfLaps;
        System.out.print(tires.get(tireType).name + "(" + laps + ")");
        if (i < bestStrategy.stints.size() - 1) {
            System.out.print(",");
        }
    }
    System.out.println();
    scanner.close();
}
}
```

<let's code/>



15

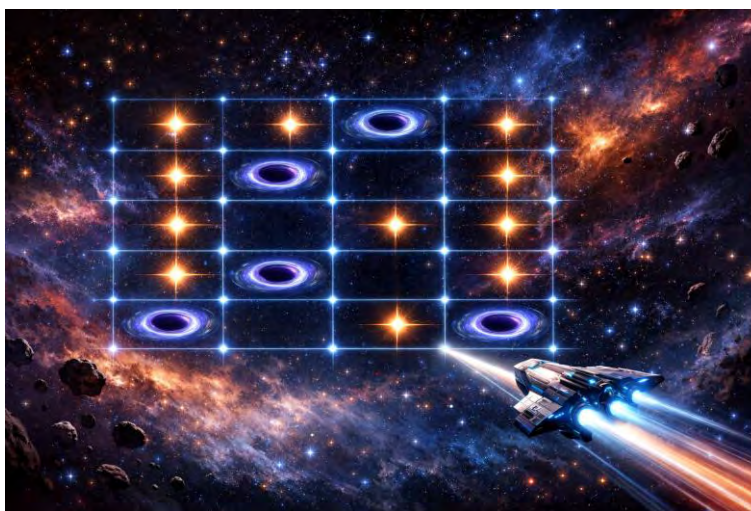
Interstellar Travel Organization

9 points

Introduction

In a distant galaxy, an advanced civilization has developed the ability to travel at the speed of light. However, their path through space is filled with stars (marked as '*') and wormholes (marked as 'O'). The distance between each star is measured in parsecs, indicated by the number preceding each star (e.g., 2* means a distance of 2 parsecs).

Each wormhole (O) halves the total travel time accumulated up to that point. The goal is to traverse the entire grid, either row by row or column by column, while minimizing travel time.



Exercise

Your task is to develop a program that:

1. Calculate the total travel time for each row and each column.
2. Determine the optimal path that minimizes the time required to travel from one edge of the grid to the opposite side.

Consider that traveling at the speed of light takes 1 year per parsec unless reduced by a wormhole. Can you find the fastest route through this cosmic maze?

Input

The input will be a galaxy represented by an array of 5x5 stars or wormholes like the following:

```
2* 2* 0 3* 2*
1* 3* 0 1* 3*
0 0 0 0 0
2* 3* 0 4* 6*
4* 5* 0 4* 5*
```

As indicated the array is filled with stars (marked as *) and wormholes (marked as O). The distance between each star is measured in parsecs, indicated by the number preceding each star (e.g., 2* means a distance of 2 parsecs).

The galaxy can only be traversed following the stars and wormholes in a row or in a column (unless it's filled by wormholes only).

The calculations in this case will be, if we follow the rows of the the array the required travel time will be:

Row	Calculation	Result
1	7.0 --> Because it's 2 + 2 , now we divide by 2 --> result is 2 + 3 + 2 = 7	7.0
2	6.0 --> Because it's 1 + 3 , now we divide by 2 --> result is 2 + 1 + 3 = 6	6.0
3	0.0 --> 0 + 0 + 0 + 0 + 0 = 0	0.0
4	12.5 --> 2 + 3 , now divide by 2 --> result is 2.5 + 4 + 6 = 12.5	12.5
5	13.5 --> ...	13.5

And for the columns:

Column	Result
1	7.5
2	10.5

```
3 0.0
4 10.0
5 13.5
```

Output

Therefore the output will be:

The best possible route would be row 2, with an interstellar space flight time of 6.0 unit

Note: If any of the values is fractional, only one decimal will be shown. This value should be rounded (4.88 would 4.9 and 4.82 would be 4.8). If the second decimal of flight time is 5 (for instance 4.25), note that the expected output would be 4.2.

Example 1

Input

```
2* 2* 0 3* 2*
1* 3* 0 1* 3*
0 0 0 0 0
2* 3* 0 4* 6*
4* 5* 0 4* 5*
```

Output

The best possible route would be row 2, with an interstellar space flight time of 6.0 unit

Example 2

Input

```
2* 2* 0 3* 0
1* 0 3* 0 3*
0 0 9* 0 1*
2* 3* 0 4* 6*
4* 5* 0 4* 5*
```

Output

The best possible route would be row 1, with an interstellar space flight time of 2.5 unit

Solutions

Python

```
def calculate_travel_time(line):
    total_time = 0
    all_wormholes = True
    for item in line:
        if item == 'O': # Wormhole found
            total_time /= 2 # Halve the current total time
        else:
            # Extract the number from the format 'X*' where X is a number
            parsec_distance = int(item[:-1]) # Remove '*' and convert to
integer
            total_time += parsec_distance
            all_wormholes = False # Found a star, so it's not all
wormholes
        # If the line was entirely wormholes, consider the time as invalid
(infinity)
        if all_wormholes:
            return float('inf') # Return a large value to indicate an invalid
path
    return total_time
def solve_galaxy(grid):
    # 5x5 grid
    rows = len(grid)
    cols = len(grid[0])
```




<CODEWARS2026/> LEÓN

```
min_time = float('inf')
min_path = None
# Calculate the travel times for each row
for i in range(rows):
    row_time = calculate_travel_time(grid[i])
    if row_time < min_time:
        min_time = row_time
        min_path = f"row {i+1}"
# Calculate the travel times for each column
for j in range(cols):
    column = [grid[i][j] for i in range(rows)]
    col_time = calculate_travel_time(column)
    if col_time < min_time:
        min_time = col_time
        min_path = f"column {j+1}"
# If the best time is infinity, it means we had an invalid path with
only wormholes.
if min_time == float('inf'):
    return "No valid path exists", 0
return min_path, min_time
# Read the galaxy grid from standard input
def read_input():
    galaxy = []
    for i in range(5):
        line = input() # Read the line directly
        galaxy.append(line.split()) # Split by spaces and append to the
galaxy list
    return galaxy
```



<CODEWARS2026/>
LEÓN

```
# Main execution
galaxy = read_input() # Input is read here, no extra print statements
# Solve the galaxy problem
best_path, best_time = solve_galaxy(galaxy)
# Print the result
if best_time == 0:
    print("No valid path exists")
else:
    print(f"The best possible route would be {best_path}, with an
interstellar space flight time of {best_time:.1f} unit")
```

C++

```
#include <iostream>
#include <vector>
#include <string>
#include <sstream>
#include <limits>
#include <iomanip>
#include <cmath>
#include <cfenv>
using namespace std;
// Function to calculate the travel time for a row or column
double calculate_travel_time(const vector<string>& line) {
    double total_time = 0;
    bool all_wormholes = true;
    for (const string& item : line) {
        if (item == "0") {
            total_time /= 2; // Halve the current total time
        } else {
```



```
        int parsec_distance = stoi(item.substr(0, item.length() - 1));
// Remove '*' and convert to integer
        total_time += parsec_distance;
        all_wormholes = false; // Found a star, so it's not all
wormholes
    }
}
// If the line was entirely wormholes, consider the time as invalid
(infinity)
    if (all_wormholes) {
        return numeric_limits<double>::infinity(); // Return a large
value to indicate an invalid path
    }
    return total_time;
}
// Function to solve the galaxy problem
pair<string, double> solve_galaxy(const vector<vector<string>>& grid) {
    int rows = grid.size();
    int cols = grid[0].size();
    double min_time = numeric_limits<double>::infinity();
    string min_path;
    // Calculate the travel times for each row
    for (int i = 0; i < rows; ++i) {
        double row_time = calculate_travel_time(grid[i]);
        if (row_time < min_time) {
            min_time = row_time;
            min_path = "row " + to_string(i + 1);
        }
    }
```



<CODEWARS2026/>
LEÓN

```
}  
// Calculate the travel times for each column  
for (int j = 0; j < cols; ++j) {  
    vector<string> column;  
    for (int i = 0; i < rows; ++i) {  
        column.push_back(grid[i][j]);  
    }  
    double col_time = calculate_travel_time(column);  
    if (col_time < min_time) {  
        min_time = col_time;  
        min_path = "column " + to_string(j + 1);  
    }  
}  
  
// If the best time is infinity, it means we had an invalid path with  
only wormholes.  
if (min_time == numeric_limits<double>::infinity()) {  
    return {"No valid path exists", 0};  
}  
return {min_path, min_time};  
}  
  
// Function to read the input  
vector<vector<string>> read_input() {  
    vector<vector<string>> galaxy(5);  
    string line;  
    for (int i = 0; i < 5; ++i) {  
        getline(cin, line);  
        stringstream ss(line);  
        string item;
```




<CODEWARS2026/>
LEÓN

```
        while (ss >> item) {
            galaxy[i].push_back(item);
        }
    }
    return galaxy;
}

// Main function
int main() {
    vector<vector<string>> galaxy = read_input(); // Read the input grid
    // Solve the galaxy problem
    auto [best_path, best_time] = solve_galaxy(galaxy);
    std::fesetround(FE_TONEAREST);              // redondeo al par más cercano
(ties-to-even)
    double print_time = std::nearbyint(best_time * 10.0) / 10.0;
    // Print the result
    if (best_time == 0) {
        cout << "No valid path exists" << endl;
    } else {
        cout << "The best possible route would be " << best_path << ",
with an interstellar space flight time of " << std::fixed <<
std::setprecision(1) << print_time << " unit" << endl;
    }
    return 0;
}
```

Java

```
import java.util.*;
import java.math.BigDecimal;
import java.math.RoundingMode;
```



<CODEWARS2026/>
LEÓN

```
public class GalaxyTravel {
    // Function to calculate the travel time for a row or column
    public static double calculateTravelTime(String[] line) {
        double totalTime = 0;
        boolean allWormholes = true;
        for (String item : line) {
            if (item.trim().equals("0")) {
                totalTime /= 2; // Halve the current total time
            } else if (item.trim().matches("[1-9]+\\*$")) {
                int parsecDistance = Integer.parseInt(item.substring(0,
item.length() - 1)); // Remove '*' and convert to integer
                totalTime += parsecDistance;
                allWormholes = false; // Found a star, so it's not all
wormholes
            }
            else
            {
                // If the line has unknown elements, consider the time as
invalid (infinity)
                return Double.POSITIVE_INFINITY;
            }
        }
        // If the line was entirely wormholes, consider the time as
invalid (infinity)
        if (allWormholes) {
            return Double.POSITIVE_INFINITY; // Return a large value to
indicate an invalid path
        }
    }
}
```



<CODEWARS2026/>
LEÓN

```
        return totalTime;
    }
    // Function to solve the galaxy problem
    public static String[] solveGalaxy(String[][] grid) {
        int rows = grid.length;
        int cols = grid[0].length;
        double minTime = Double.POSITIVE_INFINITY;
        String minPath = "";
        // Calculate the travel times for each row
        for (int i = 0; i < rows; i++) {
            double rowTime = calculateTravelTime(grid[i]);
            if (rowTime < minTime) {
                minTime = rowTime;
                minPath = "row " + (i + 1);
            }
        }
        // Calculate the travel times for each column
        for (int j = 0; j < cols; j++) {
            String[] column = new String[rows];
            for (int i = 0; i < rows; i++) {
                column[i] = grid[i][j];
            }
            double colTime = calculateTravelTime(column);
            if (colTime < minTime) {
                minTime = colTime;
                minPath = "column " + (j + 1);
            }
        }
    }
```



<CODEWARS2026/>
LEÓN

```
// If the best time is infinity, it means we had an invalid path
with only wormholes.
    if (minTime == Double.POSITIVE_INFINITY) {
        return new String[]{"No valid path exists", "0"};
    }
    BigDecimal bd = BigDecimal.valueOf(minTime).setScale(1,
RoundingMode.HALF_EVEN);
    return new String[]{minPath, bd.toPlainString()};
    //return new String[]{minPath, String.format(Locale.ENGLISH,
"%1f", minTime)};
}
// Function to read the input
public static String[][] readInput() {
    Scanner sc = new Scanner(System.in);
    String[][] galaxy = new String[5][5];
    for (int i = 0; i < 5; i++) {
        String line = sc.nextLine();
        // Split by spaces and store in the galaxy array avoiding add
empty elements
        galaxy[i] = line.split("[ ]+", 0);
    }
    return galaxy;
}
public static void main(String[] args) {
    String[][] galaxy = readInput(); // Read the input grid
    // Solve the galaxy problem
    String[] result = solveGalaxy(galaxy);
    // Print the result
```




<CODEWARS2026/>
LEÓN

```
        if (result[0].equals("No valid path exists")) {  
            System.out.println(result[0]);  
        } else {  
            System.out.println("The best possible route would be " +  
result[0] +  
                                ", with an interstellar space flight time  
of " + result[1] + " unit");  
        }  
    }  
}
```

<let's code/>



16

La cartografía del vacío

10 points

Introduction

Los Ixianos, una antigua civilización galáctica dedicada a la minería, explotación, transformación y comercio de recursos obtenidos en los anillos de asteroides, estaban maravillados con su nuevo descubrimiento: una sonda capaz de atravesar el campo de escombros que rodea a un sistema estelar sin sufrir daño alguno.

Para ellos, este hallazgo no era simplemente un avance tecnológico: representaba una posibilidad legal y cultural.

En la sociedad ixiana, interferir con mundos que albergan vida o con civilizaciones en estado primitivo es considerado un crimen contra la existencia. Por ello, hace milenios se estableció una **ley universal**:

Solo es legal extraer recursos de los cinturones de asteroides circundantes a los sistemas estelares.

De este modo, los Ixianos podían seguir prosperando sin infringir las normas que protegen el desarrollo de otras especies.

La sonda que habían encontrado tenía un nombre familiar para nosotros: **Voyager**.

Pero para los Ixianos, que abandonaron hace milenios el lenguaje hablado y escrito debido a su evolución hacia una civilización telepática, aquella estructura era algo completamente diferente: un artefacto incomprensible, lleno de símbolos, marcas y trazos cuyo propósito ignoraban.



Lo que para la humanidad era un mensaje al cosmos, para los Ixianos era un enigma visual.

Lo único que lograban “percibir” eran los vacíos, huecos y silencios entre las marcas inscritas en el artefacto.

Exercise

Como investigador humano que colabora con una delegación ixiana, debes desarrollar un Analizador Visual de Letras que permita traducir un texto humano al sistema de clasificación perceptiva de esta especie.

El programa deberá:

1. Recibir un texto (una o varias líneas).
2. Analizar cada carácter visible y clasificarlo según el tipo de vacío visual que genera:
 - o Encerrado → una cavidad completamente cerrada (o, a, b, d, e, g...) → Para los Ixianos: “camara sagrada de vacío”

- o Semiabierto → cavidad parcialmente abierta (c, m, n, r, s, w...) → Para los Ixianos: “puerta de vacío”
 - o Abierto → sin cavidad interna (i, l, t, v, x...) → Para los Ixianos: “flujo libre del cosmos”
3. Registrar línea, columna y posición absoluta de cada carácter.
 4. Generar un informe alienígena que les permita comprender la estructura del mensaje.
 5. Acompañar el programa con tests unitarios, pues los Ixianos no aceptarán un análisis defectuoso.

El listado completo de categorías de letras ixianas es el siguiente:

- **encerrado** : símbolos alfanuméricos que tienen un hueco interior cerrado
Ej.: a, b, d, e, g, o, p, q, A, B, D, O, P, Q, R, Ø, 4, 6, 8, 9
- **semiabierto_** : símbolos alfanuméricos cuya forma sugiere uno o varios huecos parcialmente abiertos.

Ej.: c, f, h, k, m, n, r, s, u, y, w, C, F, H, K, M, N, R, S, U, Y, W, E, G, 2, 3, 5)

- **abierto** : símbolos alfanuméricos que no forman hueco interior significativo.

Ej.: i, j, l, t, v, x, z, I, J, L, T, V, 16, Z, 1, 7

- **ninguno** : espacios, saltos de línea, signos de puntuación, etc.

Input

Al ejecutar el programa, el programa leera de la entrada estandar hasta encontrar una linea en blanco.

Output

Al ejecutar el programa deberá producirse un informe que represente el texto humano tal como lo percibe la mente ixiana:

no como una secuencia de fonemas, sino como un mapa geométrico de cavidades.

El objetivo final es que la delegación pueda, por primera vez en milenios, "leer" un mensaje en un lenguaje distinto al suyo.

La estructura del informe será la siguiente:

- Una primera sección llamada "Resumen Visual" con el número total de letras de cada tipo de hueco: "encerrado", "semiabierto" y "abierto". Algo así:

```
--- Resumen visual ---
Letras de hueco ENCERRADO   : 5
Letras de hueco SEMIABIERTO : 2
Letras de hueco ABIERTO    : 2
```

- Una segunda sección con el listado de letras indicando: La línea y la columna donde aparecen dentro del texto, la letra y su clasificación y el significado para los ixianos.

Sería algo así:

```
--- Detalle de letras con vacio significativo ---
Linea 1, Columna 1 | 'h' -> SEMIABIERTO | puerta de vacio
Linea 1, Columna 2 | 'o' -> ENCERRADO   | camara sagrada de vacio
Linea 1, Columna 3 | 'l' -> ABIERTO    | flujo libre del cosmos
Linea 1, Columna 4 | 'a' -> ENCERRADO   | camara sagrada de vacio
Linea 2, Columna 1 | 'a' -> ENCERRADO   | camara sagrada de vacio
Linea 2, Columna 2 | 'm' -> SEMIABIERTO | puerta de vacio
Linea 2, Columna 3 | 'i' -> ABIERTO    | flujo libre del cosmos
Linea 2, Columna 4 | 'g' -> ENCERRADO   | camara sagrada de vacio
Linea 2, Columna 5 | 'o' -> ENCERRADO   | camara sagrada de vacio
```

Example 1

Input

```
hola
amigo
```

Output

```
--- Resumen visual ---
Letras de hueco ENCERRADO   : 5
Letras de hueco SEMIABIERTO : 2
Letras de hueco ABIERTO    : 2
--- Detalle de letras con vacio significativo ---
Linea 1, Columna 1 | 'h' -> SEMIABIERTO | puerta de vacio
Linea 1, Columna 2 | 'o' -> ENCERRADO   | camara sagrada de vacio
Linea 1, Columna 3 | 'l' -> ABIERTO    | flujo libre del cosmos
Linea 1, Columna 4 | 'a' -> ENCERRADO   | camara sagrada de vacio
Linea 2, Columna 1 | 'a' -> ENCERRADO   | camara sagrada de vacio
Linea 2, Columna 2 | 'm' -> SEMIABIERTO | puerta de vacio
Linea 2, Columna 3 | 'i' -> ABIERTO    | flujo libre del cosmos
Linea 2, Columna 4 | 'g' -> ENCERRADO   | camara sagrada de vacio
Linea 2, Columna 5 | 'o' -> ENCERRADO   | camara sagrada de vacio
```

Example 2

Input

```
Bienvenido
a HP Codewars
2026
```

Output

```
--- Resumen visual ---
Letras de hueco ENCERRADO   : 13
Letras de hueco SEMIABIERTO : 9
Letras de hueco ABIERTO    : 3
--- Detalle de letras con vacio significativo ---
Linea 1, Columna 1 | 'B' -> ENCERRADO   | camara sagrada de vacio
Linea 1, Columna 2 | 'i' -> ABIERTO    | flujo libre del cosmos
Linea 1, Columna 3 | 'e' -> ENCERRADO   | camara sagrada de vacio
Linea 1, Columna 4 | 'n' -> SEMIABIERTO | puerta de vacio
Linea 1, Columna 5 | 'v' -> ABIERTO    | flujo libre del cosmos
Linea 1, Columna 6 | 'e' -> ENCERRADO   | camara sagrada de vacio
Linea 1, Columna 7 | 'n' -> SEMIABIERTO | puerta de vacio
Linea 1, Columna 8 | 'i' -> ABIERTO    | flujo libre del cosmos
Linea 1, Columna 9 | 'd' -> ENCERRADO   | camara sagrada de vacio
Linea 1, Columna 10 | 'o' -> ENCERRADO   | camara sagrada de vacio
Linea 2, Columna 1 | 'a' -> ENCERRADO   | camara sagrada de vacio
```

Línea 2, Columna 3	'H' -> SEMIABIERTO	puerta de vacío
Línea 2, Columna 4	'P' -> ENCERRADO	camara sagrada de vacío
Línea 2, Columna 6	'C' -> SEMIABIERTO	puerta de vacío
Línea 2, Columna 7	'o' -> ENCERRADO	camara sagrada de vacío
Línea 2, Columna 8	'd' -> ENCERRADO	camara sagrada de vacío
Línea 2, Columna 9	'e' -> ENCERRADO	camara sagrada de vacío
Línea 2, Columna 10	'w' -> SEMIABIERTO	puerta de vacío
Línea 2, Columna 11	'a' -> ENCERRADO	camara sagrada de vacío
Línea 2, Columna 12	'r' -> SEMIABIERTO	puerta de vacío
Línea 2, Columna 13	's' -> SEMIABIERTO	puerta de vacío
Línea 3, Columna 1	'2' -> SEMIABIERTO	puerta de vacío
Línea 3, Columna 2	'0' -> ENCERRADO	camara sagrada de vacío
Línea 3, Columna 3	'2' -> SEMIABIERTO	puerta de vacío
Línea 3, Columna 4	'6' -> ENCERRADO	camara sagrada de vacío

Solutions

Python

```
def clasificar_caracter_visual(ch):
```

```
    """
```

Clasifica un carácter según el tipo de espacio VISUAL que suele formar en una tipografía latina estándar, desde la perspectiva ixiana.

Categorías:

- "encerrado" : tiene un hueco interior cerrado
(ej.: a, b, d, e, g, o, p, q, A, B, D, O, P, Q, R, 0, 4, 6, 8, 9)
- "semiabierto" : su forma sugiere uno o varios huecos parcialmente

abiertos

(ej.: c, f, h, k, m, n, r, s, u, y, w,
C, F, H, K, M, N, R, S, U, Y, W, 2, 3, 5)

- "abierto" : no forma hueco interior significativo
(ej.: i, j, l, t, v, x, z, I, J, L, T, V, X, Z, 1, 7)
- "ninguno" : espacios, saltos de línea, signos de puntuación,

etc.

```
    """
```



<CODEWARS2026/> LEÓN

```
# Conjuntos aproximados, puramente conceptuales
encerrados = set("abdegopqABDOPQR04689")
semiabiertos = set("cfhkmnrsuywCFHKMNRSUYW235")
abiertos = set("ijltvxyzIJLTVXYZ17")
if ch in encerrados:
    return "encerrado"
elif ch in semiabiertos:
    return "semiabierto"
elif ch in abiertos:
    return "abierto"
else:
    # Por simplicidad, dejamos fuera espacios, signos, etc.
    return "ninguno"

def analizar_texto_visual(texto):
    """
    Recorre el texto y analiza cada carácter desde el punto de vista
    visual
    (espacios internos de las letras).
    Devuelve una lista de diccionarios con:
    - indice: posición absoluta en la cadena
    - linea, columna: posición en el texto (comenzando en 1)
    - caracter: el propio carácter
    - categoria: encerrado, semiabierto, abierto o ninguno
    """
    resultados = []
    linea = 1
    columna = 1
    for i, ch in enumerate(texto):
```




<CODEWARS2026/>
LEÓN

```
categoria = clasificar_caracter_visual(ch)
if categoria != "ninguno":
    resultados.append({
        "indice": i,
        "linea": linea,
        "columna": columna,
        "caracter": ch,
        "categoria": categoria
    })
# Actualizar línea y columna
if ch == "\n":
    linea += 1
    columna = 1
else:
    columna += 1
return resultados
def generar_informe_visual(texto):
    """
    Genera y muestra por pantalla un informe alienígena
    sobre los huecos visuales de las letras del texto.
    """
    datos = analizar_texto_visual(texto)
    interpretacion = {
        "encerrado": "camara sagrada de vacio",
        "semiabierto": "puerta de vacio",
        "abierto": "flujo libre del cosmos"
    }
    conteo = {
```



<CODEWARS2026/>
LEÓN

```
        "encerrado": 0,
        "semiabierto": 0,
        "abierto": 0
    }
    for d in datos:
        conteo[d["categoria"]] += 1
    print("--- Resumen visual ---")
    print(f"Letras de hueco ENCERRADO    : {conteo['encerrado']}")
    print(f"Letras de hueco SEMIABIERTO : {conteo['semiabierto']}")
    print(f"Letras de hueco ABIERTO      : {conteo['abierto']}\n")
    print("--- Detalle de letras con vacio significativo ---")
    for d in datos:
        cat = d["categoria"]
        print(
            f"Linea {d['linea']:>2}, Columna {d['columna']:>2} | "
            f"'{d['caracter']}' -> {cat.upper():11} | "
            {interpretacion[cat]}
        )
if __name__ == "__main__":
    texto_usuario = ""
    while True:
        linea = input()
        if linea.strip() == "":    # línea en blanco
            break
        texto_usuario += linea + "\n"
    texto_usuario = texto_usuario.rstrip("\n")
    generar_informe_visual(texto_usuario)
```

C++



<CODEWARS2026/> LEÓN

```
#include <iostream>
#include <string>
#include <vector>
#include <unordered_set>
#include <algorithm>
#include <iomanip>
// ----- Clasificación -----
const std::unordered_set<char> encerrados {
    'a','b','d','e','g','o','p','q',
    'A','B','D','O','P','Q','R',
    '0','4','6','8','9'
};
const std::unordered_set<char> semiabiertos {
    'c','f','h','k','m','n','r','s','u','y','w',
    'C','F','H','K','M','N','R','S','U','Y','W',
    '2','3','5'
};
const std::unordered_set<char> abiertos {
    'i','j','l','t','v','x','z',
    'I','J','L','T','V','X','Z',
    '1','7'
};
std::string clasificar_caracter_visual(char ch) {
    if (encerrados.count(ch)) return "encerrado";
    if (semiabiertos.count(ch)) return "semiabierto";
    if (abiertos.count(ch)) return "abierto";
    return "ninguno";
}
```



<CODEWARS2026/>
LEÓN

```
// ----- Estructura para resultados -----  
struct Registro {  
    int indice;  
    int linea;  
    int columna;  
    char caracter;  
    std::string categoria;  
};  
// ----- Analizar texto -----  
std::vector<Registro> analizar_texto_visual(const std::string &texto) {  
    std::vector<Registro> resultados;  
    int linea = 1;  
    int columna = 1;  
    for (int i = 0; i < (int)texto.size(); i++) {  
        char ch = texto[i];  
        std::string cat = clasificar_caracter_visual(ch);  
        if (cat != "ninguno") {  
            resultados.push_back({i, linea, columna, ch, cat});  
        }  
        if (ch == '\\n') {  
            linea++;  
            columna = 1;  
        } else {  
            columna++;  
        }  
    }  
    return resultados;  
}
```



```
// ----- Informe -----  
void generar_informe_visual(const std::string &texto) {  
    auto datos = analizar_texto_visual(texto);  
    int conteo_enc = 0;  
    int conteo_semi = 0;  
    int conteo_ab = 0;  
    for (const auto &d : datos) {  
        if (d.categoria == "encerrado") conteo_enc++;  
        else if (d.categoria == "semiabierto") conteo_semi++;  
        else if (d.categoria == "abierto") conteo_ab++;  
    }  
    std::cout << "--- Resumen visual ---\n";  
    std::cout << "Letras de hueco ENCERRADO    : " << conteo_enc << "\n";  
    std::cout << "Letras de hueco SEMIABIERTO : " << conteo_semi << "\n";  
    std::cout << "Letras de hueco ABIERTO      : " << conteo_ab << "\n\n";  
    std::cout << "--- Detalle de letras con vacio significativo ---\n";  
    for (const auto &d : datos) {  
        std::string interpretacion =  
            (d.categoria == "encerrado")    ? "camara sagrada de vacio" :  
            (d.categoria == "semiabierto") ? "puerta de vacio" :  
                                              "flujo libre del cosmos";  
        std::string categoriaMayus = d.categoria;  
        std::transform(categoriaMayus.begin(), categoriaMayus.end(),  
            categoriaMayus.begin(),  
                [](unsigned char c){ return std::toupper(c); });  
        std::cout << "Linea " << std::setw(2) << std::right << d.linea  
                    << ", Columna " << std::setw(2) << std::right <<  
d.columna
```

```
        << " | '" << d.caracter << "' -> "
        << std::left << std::setw(11) << categoriaMayus // <-- aqui
el padding
        << " | " << interpretacion
        << "\n";
    }
}
// ----- Programa principal -----
int main() {
    std::string texto_usuario;
    std::string linea;
    while (true) {
        if (!std::getline(std::cin, linea))
            break;
        // Linea en blanco -> salir
        std::string trimmed = linea;
        trimmed.erase(0, trimmed.find_first_not_of(" \t"));
        if (trimmed.empty()) break;
        texto_usuario += linea + "\n";
    }
    // Eliminar salto final si existe
    if (!texto_usuario.empty() && texto_usuario.back() == '\n')
        texto_usuario.pop_back();
    generar_informe_visual(texto_usuario);
    return 0;
}
```

Java

```
import java.util.*;
```



<CODEWARS2026/>
LEÓN

```
public class cartography {  
    // ----- Clasificación -----  
    private static final Set<Character> ENCERRADOS = new HashSet<>(  
        Arrays.asList(  
            'a','b','d','e','g','o','p','q',  
            'A','B','D','O','P','Q','R',  
            '0','4','6','8','9'  
        )  
    );  
    private static final Set<Character> SEMIABIERTOS = new HashSet<>(  
        Arrays.asList(  
            'c','f','h','k','m','n','r','s','u','y','w',  
            'C','F','H','K','M','N','R','S','U','Y','W',  
            '2','3','5'  
        )  
    );  
    private static final Set<Character> ABIERTOS = new HashSet<>(  
        Arrays.asList(  
            'i','j','l','t','v','x','z',  
            'I','J','L','T','V','X','Z',  
            '1','7'  
        )  
    );  
    public static String clasificarCaracter(char ch) {  
        if (ENCERRADOS.contains(ch)) return "encerrado";  
        if (SEMIABIERTOS.contains(ch)) return "semiabierto";  
        if (ABIERTOS.contains(ch)) return "abierto";  
        return "none";  
    }  
}
```



<CODEWARS2026/>
LEÓN

```
}  
// ----- Estructura registro -----  
public static class Registro {  
    int indice;  
    int linea;  
    int columna;  
    char character;  
    String categoria;  
    Registro(int indice, int linea, int columna, char character, String  
categoria) {  
        this.indice = indice;  
        this.linea = linea;  
        this.columna = columna;  
        this.character = character;  
        this.categoria = categoria;  
    }  
}  
// ----- Analizar texto -----  
public static List<Registro> analizarTexto(String texto) {  
    List<Registro> resultados = new ArrayList<>();  
    int linea = 1;  
    int columna = 1;  
    for (int i = 0; i < texto.length(); i++) {  
        char ch = texto.charAt(i);  
        String categoria = clasificarCaracter(ch);  
        if (!categoria.equals("none")) {  
            resultados.add(new Registro(i, linea, columna, ch,  
categoria));  
        }  
    }  
}
```




<CODEWARS2026/>
LEÓN

```
    }
    if (ch == '\n') {
        linea++;
        columna = 1;
    } else {
        columna++;
    }
}
return resultados;
}

// ----- Informe visual -----
public static void generarInforme(String texto) {
    List<Registro> datos = analizarTexto(texto);
    int encerrado = 0;
    int semiabierto = 0;
    int abierto = 0;
    for (Registro r : datos) {
        switch (r.categoria) {
            case "encerrado": encerrado++; break;
            case "semiabierto": semiabierto++; break;
            case "abierto": abierto++; break;
        }
    }
    System.out.println("--- Resumen visual ---");
    System.out.println("Letras de hueco ENCERRADO    : " + encerrado);
    System.out.println("Letras de hueco SEMIABIERTO : " +
semiabierto);
    System.out.println("Letras de hueco ABIERTO      : " + abierto);
}
```



<CODEWARS2026/>
LEÓN

```
        System.out.println();
        System.out.println("--- Detalle de letras con vacio significativo
---");
        for (Registro r : datos) {
            String interpretacion =
                r.categoria.equals("encerrado") ? "camara sagrada de
vacio" :
                r.categoria.equals("semiabierto") ? "puerta de vacio" :
                "flujo libre del
cosmos";

            System.out.printf(
                "Linea %2d, Columna %2d | '%c' -> %-11s | %s%n",
                r.linea, r.columna, r.caracter, r.categoria.toUpperCase(),
                interpretacion
            );
        }
    }
}
// ----- Main -----
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    StringBuilder sb = new StringBuilder();
    while (true) {
        if (!sc.hasNextLine()) break;
        String linea = sc.nextLine();
        if (linea.trim().isEmpty()) break;
        sb.append(linea).append("\n");
    }
    // Eliminar salto final
```



<CODEWARS2026/>
LEÓN

```
String texto = sb.toString();  
if (texto.endsWith("\n")) {  
    texto = texto.substring(0, texto.length() - 1);  
}  
generarInforme(texto);  
}  
}
```

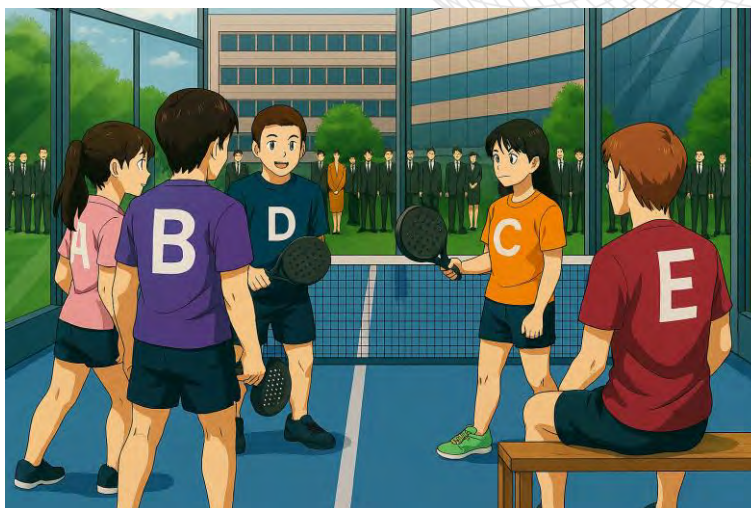
<let's code/>



17

Padel Ranking Calculator

10 points



Introduction

An IT company is organizing a padel tournament among its employees. Instead of always playing with the same pairs, the company wants employees to interact with each other and the padel ranking to be individual. Therefore, all matches resulting from the combinations of players must be played.

For example, if 5 employees sign up to the tournament (player A, B, C, D and E), player A will play with player B against all combinations of players one match. Therefore the pair AB will play against CD, against CE and against DE. And so on with all other combinations. It will, therefore, be necessary to play the following matches:

AB vs CD
AB vs DE
AB vs CE


```
AC vs BD
AC vs BE
AC vs DE
AD vs BC
AD vs BE
AD vs CE
AE vs BC
AE vs BD
AE vs CD
BC vs DE
BD vs CE
BE vs CD
```

To simplify this exercise, we'll assume that ALWAYS the tournaments have 5 players and the players are A, B, C, D, and E.

Exercise

You are asked to write a program that reads the results of the 15 matches and outputs the individual final ranking. The ranking will be ordered by the number of matches won. In case of a tie, the number of sets won will decide the winner; if there is still a tie, the number of games won and, in the extreme there is still a tie, the letter order won.

Input

The program will receive 15 lines with match results in a format like the following (please note the spacing):

```
AE vs BC:7-6, 5-7, 6-3
AB vs CE:5-7, 6-3, 7-6
AB vs DE:
AC vs BD:7-6, 5-7, 6-3
...
BE vs CD:
```

The input order can be any, but all 15 matches of the tournament will be present. Additionally, it might be possible that a match has not been played. In the previous example, match 3 has not been played.

Output

The output must be formatted as follows:

```
1- Player C - 11 Matches Won - 24 Sets Won - 135 Games Won
2- Player A - 11 Matches Won - 23 Sets Won - 139 Games Won
3- Player B - 10 Matches Won - 25 Sets Won - 119 Games Won
...
```

Example 1

Input

```
AB vs CD:7-6, 5-7, 6-3
AB vs CE:5-7, 6-3, 7-6
AB vs DE:
AC vs BD:7-6, 5-7, 6-3
AC vs BE:5-7, 6-3, 7-6
AC vs DE:
AD vs BC:7-6, 5-7, 6-3
AD vs BE:
AD vs CE:
AE vs BC:
AE vs BD:
AE vs CD:
BC vs DE:6-0, 6-1, 6-2
BD vs CE:
BE vs CD:0-6, 0-6, 0-6
```

Output

```
1- Player A - 5 Matches Won - 10 Sets Won - 90 Games Won
2- Player C - 4 Matches Won - 13 Sets Won - 120 Games Won
3- Player B - 3 Matches Won - 10 Sets Won - 102 Games Won
4- Player D - 2 Matches Won - 7 Sets Won - 71 Games Won
5- Player E - 0 Matches Won - 2 Sets Won - 35 Games Won
```

Example 2

Input

```
AB vs CD:0-6, 5-7, 6-3
AB vs CE:5-7, 6-3, 7-6
```



<CODEWARS2026/> LEÓN

```
AB vs DE:1-6, 2-6, 6-0
AC vs BD:7-6, 5-7, 6-3
AC vs BE:5-7, 6-3, 7-6
AC vs DE:6-7, 6-7, 6-7
AD vs BC:7-6, 5-7, 6-3
AD vs BE:6-7, 6-7, 6-7
AD vs CE:6-0, 6-0, 6-0
AE vs BC:5-7, 5-7, 5-7
AE vs BD:
AE vs CD:
BC vs DE:6-0, 6-1, 6-2
BD vs CE:
BE vs CD:0-6, 0-6, 0-6
```

Output

```
1- Player C - 6 Matches Won - 17 Sets Won - 159 Games Won
2- Player D - 6 Matches Won - 16 Sets Won - 140 Games Won
3- Player A - 5 Matches Won - 13 Sets Won - 161 Games Won
4- Player B - 4 Matches Won - 16 Sets Won - 146 Games Won
5- Player E - 3 Matches Won - 10 Sets Won - 104 Games Won
```

Solutions

Python

```
def parse_result(line):
    # Example: AE vs BC:7-6, 5-7, 6-3
    teams, sets = line.strip().split(":")
    team1, team2 = teams.split(" vs ")
    sets = [tuple(map(int, s.split("-"))) for s in sets.split(",")]
    return team1, team2, sets

def update_stats(player, match_won, sets_won, games_won, stats):
    stats[player]['matches'] += match_won
    stats[player]['sets'] += sets_won
    stats[player]['games'] += games_won

def main():
    players = set("ABCDE")
```



<CODEWARS2026/>
LEÓN

```
stats = {p: {'matches': 0, 'sets': 0, 'games': 0} for p in players}
for _ in range(15):
    line = input()
    _, sets = line.strip().split(":")
    if sets == "":
        continue
    team1, team2, sets = parse_result(line)
    # Validate players.
    for player in team1 + team2:
        if player not in players:
            raise ValueError("Invalid player: {}".format(player))
    # Count sets and games.
    team1_sets = team2_sets = 0
    team1_games = team2_games = 0
    for s1, s2 in sets:
        if s1 == s2 or s1 > 7 or s2 > 7 or s1 < 0 or s2 < 0:
            raise ValueError("Invalid set score: {}-{}".format(s1,
s2))

        if s1 > s2:
            team1_sets += 1
        else:
            team2_sets += 1
        team1_games += s1
        team2_games += s2
    # Determine match winner and update stats.
    winner = team1 if team1_sets > team2_sets else team2
    results = {team1: (team1 == winner, team1_sets, team1_games),
team2: (team2 == winner, team2_sets, team2_games)}
```




<CODEWARS2026/>
LEÓN

```
    for team, victories in results.items():
        match_won, sets_won, games_won = victories
        for player in team:
            update_stats(player, match_won, sets_won, games_won,
stats)
    # Sort and display ranking.
    ranking = sorted(stats.items(), key=lambda x: (-x[1]['matches'], -
x[1]['sets'], -x[1]['games'], x[0]))
    for pos, (player, data) in enumerate(ranking, 1):
        print(f"{pos}- Player {player} - {data['matches']} Matches Won -
{data['sets']} Sets Won - {data['games']} Games Won")
if __name__ == "__main__":
    main()
```

C++

```
#include <iostream>
#include <map>
#include <vector>
#include <algorithm>
#include <set>
#include <sstream>
struct Stats {
    int matches = 0, sets = 0, games = 0;
};
void updateStats(char player, int matchWon, int setsWon, int gamesWon,
std::map<char, Stats>& stats) {
    stats[player].matches += matchWon;
    stats[player].sets += setsWon;
    stats[player].games += gamesWon;
```



<CODEWARS2026/>
LEÓN

```
}  
int main() {  
    std::set<char> players = {'A','B','C','D','E'};  
    std::map<char, Stats> stats;  
    for (char player : players) stats[player] = Stats();  
    for (int i = 0; i < 15; ++i) {  
        std::string line;  
        std::getline(std::cin, line);  
        if (line.find(':') == std::string::npos || line.back() == ':')  
            continue;  
        std::string teams = line.substr(0, line.find(':'));  
        std::string setsStr = line.substr(line.find(':') + 1);  
        std::string team1 = teams.substr(0, teams.find(" vs "));  
        std::string team2 = teams.substr(teams.find(" vs ") + 4);  
        // Validate players.  
        for (char player : team1 + team2) {  
            if (players.find(player) == players.end()) {  
                throw std::invalid_argument("Invalid player: " +  
std::string(1, player));  
            }  
        }  
        // Count sets and games  
        int team1Sets = 0, team2Sets = 0, team1Games = 0, team2Games = 0;  
        std::stringstream ss(setsStr);  
        std::string set;  
        while (std::getline(ss, set, ',')) {  
            if (set.empty()) continue;  
            int s1, s2;
```

```
        sscanf(set.c_str(), "%d-%d", &s1, &s2);
        if (s1 == s2 || s1 > 7 || s2 > 7 || s1 < 0 || s2 < 0) {
            throw std::invalid_argument("Invalid set score: " +
std::to_string(s1) + "-" + std::to_string(s2));
        }
        if (s1 > s2) team1Sets++; else team2Sets++;
        team1Games += s1;
        team2Games += s2;
    }
    // Determine match winner and update stats
    std::string winner = (team1Sets > team2Sets) ? team1 : team2;
    std::vector<std::pair<std::string, std::tuple<bool, int, int>>>
results = {
    {team1, {team1 == winner, team1Sets, team1Games}},
    {team2, {team2 == winner, team2Sets, team2Games}}
};
    for (const auto& [team, victories] : results) {
        auto [won, sets_won, games_won] = victories;
        for (char player : team) {
            updateStats(player, won ? 1 : 0, sets_won, games_won,
stats);
        }
    }
    // Sort and display ranking.
    std::vector<std::pair<char, Stats>> ranking(stats.begin(),
stats.end());
```



<CODEWARS2026/>
LEÓN

```
std::sort(ranking.begin(), ranking.end(), [](const auto& a, const
auto& b) {
    if (b.second.matches != a.second.matches)
        return b.second.matches < a.second.matches;
    if (b.second.sets != a.second.sets)
        return b.second.sets < a.second.sets;
    if (b.second.games != a.second.games)
        return b.second.games < a.second.games;
    return a.first < b.first;
});
int pos = 1;
for (const auto& entry : ranking) {
    const Stats& d = entry.second;
    printf("%d- Player %c - %d Matches Won - %d Sets Won - %d Games
Won\n",
        pos++, entry.first, d.matches, d.sets, d.games);
}
return 0;
}
```

Java

```
import java.util.*;
public class PadelRanking {
    static class Stats {
        int matches = 0, sets = 0, games = 0;
    }
    static void updateStats(char player, int matchWon, int setsWon, int
gamesWon, Map<Character, Stats> stats) {
        Stats s = stats.get(player);
```




<CODEWARS2026/>
LEÓN

```
s.matches += matchWon;
s.sets += setsWon;
s.games += gamesWon;
}
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    Set<Character> players = new
HashSet<>(Arrays.asList('A','B','C','D','E'));
    Map<Character, Stats> stats = new HashMap<>();
    for (char player : players) stats.put(player, new Stats());
    for (int i = 0; i < 15; i++) {
        String line = sc.nextLine().trim();
        if (!line.contains(":") || line.endsWith(":")) continue;
        String[] parts = line.split(":");
        String[] teams = parts[0].split(" vs ");
        String team1 = teams[0], team2 = teams[1];
        // Validate players.
        String allPlayers = team1 + team2;
        for (char player : allPlayers.toCharArray()) {
            if (!players.contains(player)) {
                throw new IllegalArgumentException("Invalid player: "
+ player);
            }
        }
        // Count sets and games.
        String[] setsStr = parts[1].split(",");
        int team1Sets = 0, team2Sets = 0, team1Games = 0, team2Games =
0;
```

```
for (String set : setsStr) {
    set = set.trim();
    if (set.isEmpty()) continue;
    String[] gamesWon = set.split("-");
    int s1 = Integer.parseInt(gamesWon[0]);
    int s2 = Integer.parseInt(gamesWon[1]);
    if (s1 == s2 || s1 > 7 || s2 > 7 || s1 < 0 || s2 < 0) {
        throw new IllegalArgumentException("Invalid set score:
" + s1 + "-" + s2);
    }
    if (s1 > s2) team1Sets++; else team2Sets++;
    team1Games += s1;
    team2Games += s2;
}

// Determine match winner and update stats.
String winner = (team1Sets > team2Sets) ? team1 : team2;
Map<String, int[]> results = new HashMap<>();
results.put(team1, new int[]{team1.equals(winner) ? 1 : 0,
team1Sets, team1Games});
results.put(team2, new int[]{team2.equals(winner) ? 1 : 0,
team2Sets, team2Games});
for (Map.Entry<String, int[]> entry : results.entrySet()) {
    String team = entry.getKey();
    int[] victories = entry.getValue();
    int matchWon = victories[0];
    int setsWon = victories[1];
    int gamesWon = victories[2];
    for (char player : team.toCharArray()) {
```

```
        updateStats(player, matchWon, setsWon, gamesWon,
stats);
    }
}
// Sort and display ranking.
List<Map.Entry<Character, Stats>> ranking = new
ArrayList<>(stats.entrySet());
ranking.sort((a, b) -> {
    if (b.getValue().matches != a.getValue().matches)
        return b.getValue().matches - a.getValue().matches;
    if (b.getValue().sets != a.getValue().sets)
        return b.getValue().sets - a.getValue().sets;
    if (b.getValue().games != a.getValue().games)
        return b.getValue().games - a.getValue().games;
    return a.getKey() - b.getKey();
});
int pos = 1;
for (Map.Entry<Character, Stats> entry : ranking) {
    Stats d = entry.getValue();
    System.out.printf("%d- Player %c - %d Matches Won - %d Sets
Won - %d Games Won\n",
        pos++, entry.getKey(), d.matches, d.sets, d.games);
}
}
```


18

Roland Garros Point Counter

11 points

Introduction

You are still working on the Roland Garros project, and it turns out that the person in charge of programming the match scoreboard has left the company, leaving their part of the project unfinished.

Since you did a great job with the Tie-Break scoring, you have been assigned the development of the match scoreboard.



As with the Tie-Break program, the AI analyzes the game and detects who won each point. And as we all know, computers work with 0s and 1s, so the AI simply sends us zeros and ones.

Specifically, it sends us a string of zeros and ones separated by commas, just like in the Tie-Break, where 0 means that “Player 1” won the point and 1 means that “Player 2” won the point. Careful, don’t get confused: a “1” does NOT mean Player 1 won the point.

The scoring in a game is 0, 15, 30, 40. If it reaches 40-40, the player who wins the next point has “advantage,” which we will call “Adv.” If the player with advantage loses the point, it goes back to 40-40. But if they win it, they win the game. A “set” is won when a player reaches 6 games, provided they have a two-game lead. For example, a set can be won if a player wins the sixth game and the score is 6-4. But if it’s 6-5, the set continues and could end 7-5 or 6-6. If it ends 6-6, that’s when the Tie-Break is played. Since you already programmed that part in a previous exercise, and to simplify, we will not address Tie-Breaks here. That means if a set ends 6-6, it will appear as such on the scoreboard.

Exercise

You are asked to develop a program that receives a list of zeros and ones corresponding to Players 1 and 2 and tells us the current match score.

Note that the program is still in the testing phase and has a couple of issues:

- Points played during Tie-Breaks are not counted. If a game ends in a Tie-Break, we will show it as 6-6 and not add a winner (see example 3).
- The program does not yet detect the end of the match; more than 5 sets could appear.

Input

The input will be a list of zeros and ones. For example:

```
1,1,0,1,1,0,1,0,0,0,1,1,0,0,0,1,1,0,0,0,0,1,1,0,1,0,1,0,1,1,1,0,0,0,1,1,0,1,1,0,1,1,0,1,1,0,1,0,0,0,1,1,0,0,0,1,1,0,0,0,0,1,1,0,0,0,0,1,1,0,1,0,1,0,1,1,1,0,0,0,1,1,0
```

Output

The program will generate an output with the following format (in spanish):

Jugador 1: <sets_ganados> <Juegos_ganados_set_actual> <Puntos_juego_actual>

Jugador 2: <sets_ganados> <Juegos_ganados_set_actual> <Puntos_juego_actual>

Sets terminados <set1>,<set2>,<set3>

For instance, with the given input:

```
Jugador 1: 0 4 Adv  
Jugador 2: 0 4 40  
Sets terminados:
```

Note that even when there are no "Set terminados", there's an space AFTER THE COLON.

Example1

Input

```
1,1,0,1,1,0,1,0,0,0,1,1,0,0,0,1,1,0,0,0,0,1,1,0,1,0,1,0,1,1,1,0,0,0,1,1,0,1,1,0,1,  
1,0,1,0,0,0,1,1,0,0,0,1,1,0,0,0,0,1,1,0,1,0,1,0,1,1,1,0,0,0,1,1,0,0,0,1,1,0
```

Output

```
Jugador 1: 0 4 Adv  
Jugador 2: 0 4 40  
Sets terminados:
```

Example2

Input

```
1,1,0,1,1,0,1,0,0,0,1,1,0,0,0,1,1,0,0,0,0,1,1,0,1,0,1,0,1,1,1,0,0,0,1,1,0,1,1,0,1,  
1,0,1,0,0,0,1,1,0,0,0,1,1,0,0,0,0,1,1,0,1,0,1,0,1,1,1,0,0,0,1,1,0,1,1,1,0,0,0,1,1,  
0,0,0,0,1,1,0,1,0,1,0,1,1,1,0,0,0,1,1,0,1,1,0,1,1,0,1,1,0,1,0,0,0,1,1,0,0,0,1,1,0,  
1,1,0,1,0,1,0,1,1,1,0,0,0,1,1,0,1,1,0,1,1,0,1,0,0,0,1,1,0,0,0,1,1,0,0,0,1,1,0,1,  
0,1,0,1,1,1,0,0,0
```

Output

```
Jugador 1: 0 4 40  
Jugador 2: 0 5 15  
Sets terminados: 6-6
```

Example3

Input

```
1,1,0,1,1,0,1,0,0,0,1,1,0,0,0,1,1,0,0,0,0,1,1,0,1,0,1,0,1,1,1,0,0,0,1,1,0,1,1,0,1,
1,0,1,0,0,0,1,1,0,0,0,1,1,0,0,0,0,1,1,0,1,0,1,0,1,1,1,0,0,0,1,1,0,1,1,0,0,0,1,1,
0,0,0,0,1,1,0,1,0,1,0,1,1,1,0,0,0,1,1,0,1,1,0,1,1,0,1,1,0,0,0,1,1,0,0,0,1,1,0,
1,1,0,1,0,1,0,1,1,1,0,0,0,1,1,0,1,1,0,1,1,0,1,1,0,1,0,0,0,1,1,0,0,0,1,1,0,
0,1,0,1,1,1,0,0,0,1,1,1,1,0,0,0,1,1,1,1,0,0,0,1,1,0,0,0,0,1,1,0,1,
0,1,0,1,1,1,0,0,0,0,1,1,1,1,1,0,0,0,0
```

Output

```
Jugador 1: 0 0 15
Jugador 2: 0 0 0
Sets terminados: 6-6,6-6
```

Solutions

Python

global ventaja

```
def marcador_tenis(puntos):
    puntuacion = [0, 15, 30, 40]
    sets = [0, 0]      # Sets ganados por cada jugador
    juegos = [0, 0]    # Juegos en el set actual
    puntos_actuales = [0, 0]
    ventaja = None
    sets_terminados = [] # Lista para guardar sets finalizados
    for punto in puntos:
        #ganador y perdedor son 0 o 1
        ganador = punto
        perdedor = 1 - ganador
        # Si hay ventaja
        if ventaja is not None:
            if ganador == ventaja:
```

```
juegos[ganador] += 1
ventaja = None
puntos_actuales = [0, 0]
else:
    ventaja = None
    puntos_actuales[ganador] += 1
else:
    puntos_actuales[ganador] += 1
    #print (str(puntos_actuales[0])+"-"+str(puntos_actuales[1]))
    #1 --> 15, 2--> 30, 3 --> 40 , 4--> Juego o Adv. Se puede
    ganar con 4, si el perdedor solo tiene 2 (30). 0 con 5, si el
    #Perdedor solo tiene 3 (40)
    if puntos_actuales[ganador] >= 4 and puntos_actuales[ganador]
- puntos_actuales[perdedor] >= 2:
        juegos[ganador] += 1
        #Reseteamos los puntos del juego
        puntos_actuales = [0, 0]
        #Estamos en 40-40 o en Adv-40
        elif puntos_actuales[0] >= 3 and puntos_actuales[1] >= 3:
            if puntos_actuales[ganador] > puntos_actuales[perdedor]:
                ventaja = ganador

    #print (str(punto)+"--> Estamos "+str(juegos[0])+"-
"+str(juegos[1])+" y "+(str(puntos_actuales[0])+"-"+
str(puntos_actuales[1]) ))
    # Comprobar si alguien gana el set
    if juegos[ganador] >= 6 and juegos[ganador] - juegos[perdedor] >=
2:

        sets[ganador] += 1
```



```
sets_terminados.append(f"{juegos[0]}-{juegos[1]}")
juegos = [0, 0]
puntos_actuales = [0, 0]
ventaja = None
elif juegos[0] == 6 and juegos[1] == 6:
    # Set termina en 6-6 (sin tie-break)
    sets_terminados.append("6-6")
    juegos = [0, 0]
    puntos_actuales = [0, 0]
    ventaja = None
# Determinar puntos actuales para mostrar
def puntos_texto(ventaja,puntos_actuales):
    if ventaja is not None:
        return ("Adv" if ventaja == 0 else "40", "Adv" if ventaja == 1
else "40")
    else:
        return (
            puntuacion[min(puntos_actuales[0], 3)],
            puntuacion[min(puntos_actuales[1], 3)]
        )
p1, p2 = puntos_texto(ventaja, puntos_actuales)
# Mostrar marcador
print(f"Jugador 1: {sets[0]} {juegos[0]} {p1}")
print(f"Jugador 2: {sets[1]} {juegos[1]} {p2}")
# Mostrar sets terminados
print("Sets terminados: " + ",".join(sets_terminados))
# Leer entrada
entrada = input().strip()
```



<CODEWARS2026/>
LEÓN

```
puntos = [int(x) for x in entrada.split(",")]  
marcador_tenis(puntos)
```

C++

```
#include <iostream>  
#include <vector>  
#include <sstream>  
#include <string>  
using namespace std;  
void marcadorTennis(const vector<int>& puntos) {  
    int puntuacion[] = {0, 15, 30, 40};  
    int sets[2] = {0, 0};  
    int juegos[2] = {0, 0};  
    int puntosActuales[2] = {0, 0};  
    int ventaja = -1;  
    vector<string> setsTerminados;  
    for (int punto : puntos) {  
        int ganador = punto;  
        int perdedor = 1 - ganador;  
        if (ventaja != -1) {  
            if (ganador == ventaja) {  
                juegos[ganador]++;  
                ventaja = -1;  
                puntosActuales[0] = puntosActuales[1] = 0;  
            } else {  
                ventaja = -1;  
                puntosActuales[ganador]++;  
            }  
        } else {
```

```
puntosActuales[ganador]++;  
if (puntosActuales[ganador] >= 4 && puntosActuales[ganador] -  
puntosActuales[perdedor] >= 2) {  
    juegos[ganador]++;  
    puntosActuales[0] = puntosActuales[1] = 0;  
} else if (puntosActuales[0] >= 3 && puntosActuales[1] >= 3) {  
    if (puntosActuales[ganador] > puntosActuales[perdedor]) {  
        ventaja = ganador;  
    }  
}  
}  
if (juegos[ganador] >= 6 && juegos[ganador] - juegos[perdedor] >=  
2) {  
    sets[ganador]++;  
    setsTerminados.push_back(to_string(juegos[0]) + "-" +  
to_string(juegos[1]));  
    juegos[0] = juegos[1] = 0;  
    puntosActuales[0] = puntosActuales[1] = 0;  
    ventaja = -1;  
} else if (juegos[0] == 6 && juegos[1] == 6) {  
    setsTerminados.push_back("6-6");  
    juegos[0] = juegos[1] = 0;  
    puntosActuales[0] = puntosActuales[1] = 0;  
    ventaja = -1;  
}  
}  
string p1, p2;  
if (ventaja != -1) {
```



<CODEWARS2026/>
LEÓN

```
        p1 = (ventaja == 0) ? "Adv" : "40";
        p2 = (ventaja == 1) ? "Adv" : "40";
    } else {
        p1 = to_string(puntuacion[min(puntosActuales[0], 3)]);
        p2 = to_string(puntuacion[min(puntosActuales[1], 3)]);
    }
    cout << "Jugador 1: " << sets[0] << " " << juegos[0] << " " << p1 <<
endl;
    cout << "Jugador 2: " << sets[1] << " " << juegos[1] << " " << p2 <<
endl;
    cout << "Sets terminados: ";
    for (size_t i = 0; i < setsTerminados.size(); i++) {
        cout << setsTerminados[i];
        if (i < setsTerminados.size() - 1) cout << ",";
    }
    cout << endl;
}

int main() {
    string entrada;
    getline(cin, entrada);
    stringstream ss(entrada);
    string token;
    vector<int> puntos;
    while (getline(ss, token, ',')) {
        puntos.push_back(stoi(token));
    }
    marcadorTenis(puntos);
    return 0;
}
```




<CODEWARS2026/>
LEÓN

```
}
```

Java

```
import java.util.*;

public class RolandGarrosPointCounter {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String entrada = sc.nextLine().trim();
        String[] partes = entrada.split(",");
        int[] puntos = new int[partes.length];
        for (int i = 0; i < partes.length; i++) {
            puntos[i] = Integer.parseInt(partes[i].trim());
        }
        marcadorTenis(puntos);
    }

    public static void marcadorTenis(int[] puntos) {
        int[] puntuacion = {0, 15, 30, 40};
        int[] sets = {0, 0};
        int[] juegos = {0, 0};
        int[] puntosActuales = {0, 0};
        Integer ventaja = null;
        List<String> setsTerminados = new ArrayList<>();
        for (int punto : puntos) {
            int ganador = punto;
            int perdedor = 1 - ganador;
            if (ventaja != null) {
                if (ganador == ventaja) {
                    juegos[ganador]++;
                    ventaja = null;
                }
            }
        }
    }
}
```

<let's code/>



```
        puntosActuales[0] = puntosActuales[1] = 0;
    } else {
        ventaja = null;
        puntosActuales[ganador]++;
    }
} else {
    puntosActuales[ganador]++;
    if (puntosActuales[ganador] >= 4 &&
puntosActuales[ganador] - puntosActuales[perdedor] >= 2) {
        juegos[ganador]++;
        puntosActuales[0] = puntosActuales[1] = 0;
    } else if (puntosActuales[0] >= 3 && puntosActuales[1] >=
3) {
        if (puntosActuales[ganador] >
puntosActuales[perdedor]) {
            ventaja = ganador;
        }
    }
}
if (juegos[ganador] >= 6 && juegos[ganador] - juegos[perdedor]
>= 2) {
    sets[ganador]++;
    setsTerminados.add(juegos[0] + "-" + juegos[1]);
    juegos[0] = juegos[1] = 0;
    puntosActuales[0] = puntosActuales[1] = 0;
    ventaja = null;
} else if (juegos[0] == 6 && juegos[1] == 6) {
    setsTerminados.add("6-6");
}
```

```
juegos[0] = juegos[1] = 0;
puntosActuales[0] = puntosActuales[1] = 0;
ventaja = null;
    }
}
String p1, p2;
if (ventaja != null) {
    p1 = (ventaja == 0) ? "Adv" : "40";
    p2 = (ventaja == 1) ? "Adv" : "40";
} else {
    p1 = String.valueOf(puntuacion[Math.min(puntosActuales[0],
3))]);
    p2 = String.valueOf(puntuacion[Math.min(puntosActuales[1],
3))]);
}
System.out.println("Jugador 1: " + sets[0] + " " + juegos[0] + "
" + p1);
System.out.println("Jugador 2: " + sets[1] + " " + juegos[1] + "
" + p2);
System.out.println("Sets terminados: " + String.join(",",
setsTerminados));
}
}
```


19

Un problema universal

11 points

Introduction

Cuando los Tricachfin descubrieron a los habitantes del planeta de las tres lunas, la galaxia se estremeció. No por el hallazgo –había millones de mundos primitivos–, sino porque decidieron hablarles, saltándose el Protocolo de Silencio Cósmico.

El Protocolo de Silencio Cósmico lo prohibía con absoluta claridad:

“Ninguna mente que controle la energía de un universo debe alterar el destino de una mente que aún no controla la energía de su propio hogar.”

...Pero la curiosidad es un instinto más antiguo que las leyes.

El equipo de enlace –formado por entidades sin forma, que viajaban entre dimensiones del pensamiento– descendió sobre el pequeño mundo azul-verde. Allí ,habitaban los N’mari, una especie que apenas había aprendido a lanzar fragmentos de sí mismos al vacío del espacio.



El problema apareció pronto. Los N'mari no entendían las ecuaciones. No porque fueran simples, sino porque sus matemáticas eran un arte, no una ciencia. Sus sacerdotes pregonaban que sumar y multiplicar eran actos sagrados, gestos de expansión, mientras que restar y dividir eran rituales de destrucción. En su escritura numérica, el orden de los signos dependía del sentimiento: si una ecuación "lucía fea", debía rehacerse hasta que resultara hermosa.

Los Tricachfin intentaron enseñarles el lenguaje universal: las matemáticas puras. Pero cada fórmula enviada a su red de cómputo planetario era reinterpretada con lógica estética. La ecuación de propulsión cuántica, por ejemplo, se transformó en un poema sobre el fuego y el vacío. El modelo de curvatura gravitatoria terminó como una plegaria a sus dioses binarios. En resumen, era como intentar enseñarles matemáticas puras a una gallina.

Así que decidieron buscar un traductor entre civilizaciones: Tú.

Exercise

Como traductor entre ambas civilizaciones, tu misión es construir un intérprete que evalúe expresiones matemáticas tal como lo haría la civilización de los N'mari. Tu tarea es escribir un programa que, dada una cadena con una operación, devuelva su resultado según las reglas estéticas de su mundo.

Reglas de la aritmética estética

- 1.- Los operadores $+$ y $*$ poseen mayor jerarquía: se calculan antes que $-$ y $/$.
- 2.- Las operaciones del mismo grupo se resuelven de izquierda a derecha.
- 3.- No existen paréntesis (son considerados herejía simbólica).
- 4.- Se admiten números enteros o decimales, y signos unarios ($+$ o $-$ al inicio).

La civilización de los N'mari aborrecen las divisiones por cero –ya esto causaría una catástrofe galáctica (ZeroDivisionError). Y si la expresión es ilegible o contradictoria, lanza un ValueError, pues incluso en su universo, el sinsentido no tiene cabida.

Input

El programa recibirá una única línea con una fórmula matemática que cumpla las reglas anteriores. Por ejemplo:

```
3+4*2
```

Output

El programa mostrará el resultado según las reglas de la aritmética estética. Por ejemplo, con la anterior entrada, el resultado en matemática estándar será: 11. Pero para la Civilización N'mari, primero van la suma y la multiplicación (+ y *) y dentro de ellas, el orden (de izquierda a derecha): $3+4=7$, luego $7*2=14$.

El programa simplemente mostrará el resultado:

```
14
```

Nota: Por simplicidad, solo se trabajará con decimales de un único dígito. Los números enteros se mostrarán sin decimales.

Nota 2: Podría ocurrir que la entrada simplemente reciba un número sin ningún tipo de operación, la salida será ese mismo número

Nota 3: Si en algún momento se produce una división por cero, se mostrará por pantalla "ZeroDivisionError"

Nota 4: Y si se produce una entrada inválida: "3++4", "2a+1", "3*(4+2)", se escribirá "ValueError"... recordemos que no hay paréntesis en la civilización N'mari



<CODEWARS2026/> LEÓN

Example 1

Input

8-2/2

Output

3

Ya que no hay + ni *, por la estética N'mari se resuelven - y / al mismo nivel, de izquierda a derecha: $8-2=6$, $6/2=3$

Example 2

Input

10/2+3*2-4

Output

-3

Example 3

Input

-3 + 4 * 2

Output

2

Example 4

Input

2.5*2+1.5



<CODEWARS2026/> LEÓN

Output

6.5

Example 5

Input

42

Output

42

Example 6

Input

1+2/0

Output

ZeroDivisionError

Example 7

Input

3+(4-45)

Output

ValueError

Solutions

Python



<CODEWARS2026/>
LEÓN

```
import re
from typing import Union, List
Number = Union[int, float]
class Token:
    def __init__(self, kind: str, value: str):
        self.kind = kind # 'num' or 'op'
        self.value = value
    def __repr__(self):
        return f"Token({self.kind!r}, {self.value!r})"
def _to_number(s: str) -> Number:
    # Devuelve int si es exacto, si no float
    if '.' in s:
        val = float(s)
        return int(val) if val.is_integer() else val
    return int(s)
def _tokenize(expr: str) -> List[Token]:
    """
    Tokeniza soportando signo unario +/- . No admite paréntesis.
    Lanza ValueError si encuentra caracteres inválidos.
    """
    expr = expr.strip()
    if any(c in expr for c in '()[{}'):
        raise ValueError("Paréntesis no permitidos en la notación N'mari")
    tokens: List[Token] = []
    i = 0
    n = len(expr)
    def is_op(ch: str) -> bool:
        return ch in '+-*/'
```



<CODEWARS2026/>
LEÓN

```
while i < n:
    ch = expr[i]
    if ch.isspace():
        i += 1
        continue
    # Número (posible signo unario)
    if ch.isdigit() or ch == '.' or (is_op(ch) and (ch in '+-')):
        # Detectar si es signo unario: al inicio o tras otro operador
        if (ch in '+-') and (i == 0 or (tokens and tokens[-1].kind ==
'op') or (not tokens)):
            # signo unario
            j = i + 1
            num_str = ch
            # deben venir dígitos o punto
            if j < n and (expr[j].isdigit() or expr[j] == '.'):
                while j < n and (expr[j].isdigit() or expr[j] == '.'):
                    num_str += expr[j]
                    j += 1
            # validar formato numérico
            if num_str in ['+', '-'] or num_str.count('.') > 1:
                raise ValueError("Número mal formado")
            tokens.append(Token('num', num_str))
            i = j
            continue
        # Si no venían dígitos, entonces no era unario válido
        # caemos a tratarlos como operador normal más adelante
    # número sin signo unario explícito
    if ch.isdigit() or ch == '.':
```

```
j = i
num_str = ''
dot_count = 0
while j < n and (expr[j].isdigit() or expr[j] == '.'):
    if expr[j] == '.':
        dot_count += 1
        if dot_count > 1:
            raise ValueError("Número mal formado")
    num_str += expr[j]
    j += 1
if num_str in ['.', '']:
    raise ValueError("Número mal formado")
tokens.append(Token('num', num_str))
i = j
continue

# Operador binario
if is_op(ch):
    tokens.append(Token('op', ch))
    i += 1
    continue

# Cualquier otro carácter es inválido
raise ValueError(f"Carácter inválido: {ch!r}")

# Validaciones simples de secuencia
if not tokens:
    raise ValueError("Expresión vacía")
if tokens[0].kind == 'op' and tokens[0].value not in ['+', '-']:
    raise ValueError("Expresión no puede iniciar con operador binario")
```



<CODEWARS2026/>
LEÓN

```
if tokens[-1].kind == 'op':
    raise ValueError("Expresión no puede terminar con operador")
# No permitir dos operadores binarios seguidos (salvo que el segundo
se
# hubiese absorbido como signo unario, lo cual ya tratamos arriba)
for a, b in zip(tokens, tokens[1:]):
    if a.kind == 'op' and b.kind == 'op':
        raise ValueError("Operadores consecutivos no permitidos")
return tokens
def _to_rpn(tokens: List[Token]) -> List[Token]:
    """
    Shunting-yard con precedencia estética:
    + y * : precedencia alta (2)
    - y / : precedencia baja (1)
    Asociatividad izquierda.
    """
    prec = {'+': 2, '*': 2, '-': 1, '/': 1}
    out: List[Token] = []
    ops: List[Token] = []
    for t in tokens:
        if t.kind == 'num':
            out.append(t)
        else: # operador
            while ops and ops[-1].kind == 'op' and prec[ops[-1].value] >=
prec[t.value]:
                out.append(ops.pop())
            ops.append(t)
    while ops:
```




<CODEWARS2026/>
LEÓN

```
        out.append(ops.pop())
    return out

def _eval_rpn(rpn: List[Token]) -> Number:
    stack: List[float] = []
    for t in rpn:
        if t.kind == 'num':
            stack.append(float(t.value))
        else:
            if len(stack) < 2:
                raise ValueError("ValueError")
            b = stack.pop()
            a = stack.pop()
            op = t.value
            if op == '+':
                stack.append(a + b)
            elif op == '-':
                stack.append(a - b)
            elif op == '*':
                stack.append(a * b)
            elif op == '/':
                if b == 0:
                    raise ZeroDivisionError("ZeroDivisionError")
                stack.append(a / b)
            else:
                raise ValueError(f"Operador desconocido: {op}")
    if len(stack) != 1:
        raise ValueError("ValueError")
    res = stack[0]
```

<let's code/>





<CODEWARS2026/> LEÓN

```
return int(res) if float(res).is_integer() else res
def evaluate_kardashev(expr: str) -> Number:
    """
    Evalúa 'expr' con la precedencia de la civilización N'mari:
    prioridad alta: +, *
    prioridad baja: -, /
    Todas las operaciones son asociativas por la izquierda.
    """
    tokens = _tokenize(expr)
    rpn = _to_rpn(tokens)
    return _eval_rpn(rpn)
data=input("")
try:
    print(evaluate_kardashev(data))
except ZeroDivisionError:
    print("ZeroDivisionError")
except ValueError:
    print("ValueError")
```

```
C++
#include <bits/stdc++.h>
using namespace std;
struct Token {
    string kind; // "num" o "op"
    string value;
};
double toNumber(const string &s) {
    if (s.find('.') != string::npos) {
        double val = stod(s);
```



<CODEWARS2026/>
LEÓN

```
        return val;
    }
    return stoi(s);
}

vector<Token> tokenize(const string &expr) {
    string e = expr;
    if (e.find('(') != string::npos || e.find(')') != string::npos)
        throw runtime_error("ValueError");
    vector<Token> tokens;
    int i = 0, n = e.size();
    auto isOp = [](char ch) {
        return ch == '+' || ch == '-' || ch == '*' || ch == '/';
    };
    while (i < n) {
        char ch = e[i];
        if (isspace(ch)) { i++; continue; }
        if (isdigit(ch) || ch == '.' || (isOp(ch) && (ch == '+' || ch == '-' || ch == '* || ch == '/'))) {
            if ((ch == '+' || ch == '-') && (i == 0 || (!tokens.empty() && tokens.back().kind == "op"))) {
                int j = i + 1;
                string numStr(1, ch);
                if (j < n && (isdigit(e[j]) || e[j] == '.')) {
                    while (j < n && (isdigit(e[j]) || e[j] == '.')) {
                        numStr += e[j++];
                    }
                    tokens.push_back({"num", numStr});
                    i = j;
                }
            }
        }
    }
}
```



<CODEWARS2026/>
LEÓN

```
        continue;
    }
}
if (isdigit(ch) || ch == '.') {
    int j = i;
    string numStr;
    int dotCount = 0;
    while (j < n && (isdigit(e[j]) || e[j] == '.')) {
        if (e[j] == '.') {
            dotCount++;
            if (dotCount > 1) throw
runtime_error("ValueError");
        }
        numStr += e[j++];
    }
    tokens.push_back({"num", numStr});
    i = j;
    continue;
}
if (isOp(ch)) {
    tokens.push_back({"op", string(1, ch)});
    i++;
    continue;
}
throw runtime_error(string("ValueError") + ch);
}
if (tokens.empty()) throw runtime_error("ValueError");
```

<let's code/>





```
        if (tokens.front().kind == "op" && tokens.front().value != "+" &&
tokens.front().value != "-")
            throw runtime_error("ValueError");
        if (tokens.back().kind == "op")
            throw runtime_error("ValueError");
        for (size_t k = 0; k + 1 < tokens.size(); k++) {
            if (tokens[k].kind == "op" && tokens[k + 1].kind == "op")
                throw runtime_error("ValueError");
        }
        return tokens;
    }
    vector<Token> toRPN(const vector<Token> &tokens) {
        map<string, int> prec = {{ "+", 2}, {"*", 2}, {"-", 1}, {"/", 1}};
        vector<Token> out;
        stack<Token> ops;
        for (auto &t : tokens) {
            if (t.kind == "num") out.push_back(t);
            else {
                while (!ops.empty() && prec[ops.top().value] >= prec[t.value])
                {
                    out.push_back(ops.top());
                    ops.pop();
                }
                ops.push(t);
            }
        }
        while (!ops.empty()) {
            out.push_back(ops.top());
        }
    }
}
```



<CODEWARS2026/>
LEÓN

```
        ops.pop();
    }
    return out;
}

double evalRPN(const vector<Token> &rpn) {
    stack<double> st;
    for (auto &t : rpn) {
        if (t.kind == "num") st.push(stod(t.value));
        else {
            if (st.size() < 2) throw runtime_error("ValueError");
            double b = st.top(); st.pop();
            double a = st.top(); st.pop();
            if (t.value == "+") st.push(a + b);
            else if (t.value == "-") st.push(a - b);
            else if (t.value == "*") st.push(a * b);
            else if (t.value == "/") {
                if (b == 0) throw runtime_error("ZeroDivisionError");
                st.push(a / b);
            }
        }
    }
    if (st.size() != 1) throw runtime_error("ValueError");
    return st.top();
}

int main() {
    string expr;
    getline(cin, expr);
    try{
```



<CODEWARS2026/>
LEÓN

```
    auto tokens = tokenize(expr);
    auto rpn = toRPN(tokens);
    double res = evalRPN(rpn);
    if (floor(res) == res) cout << (int)res << "\n";
    else cout << res << "\n";
} catch (const runtime_error &e) {
    cout << e.what() << "\n";
}
return 0;
}
```

Java

```
import java.util.*;
import java.io.*;
class Token {
    String kind; // "num" o "op"
    String value;
    Token(String k, String v) { kind = k; value = v; }
}
public class Un_problema_universal {
    static List<Token> tokenize(String expr) {
        expr = expr.trim();
        if (expr.contains("(") || expr.contains(")"))
            throw new RuntimeException("ValueError");
        List<Token> tokens = new ArrayList<>();
        int i = 0, n = expr.length();
        while (i < n) {
            char ch = expr.charAt(i);
            if (Character.isWhitespace(ch)) { i++; continue; }
```



<CODEWARS2026/>
LEÓN

```
        if (Character.isDigit(ch) || ch == '.' || (isOp(ch) && (ch ==
'+' || ch == '-')))) {
            if ((ch == '+' || ch == '-') && (i == 0 ||
(!tokens.isEmpty() && tokens.get(tokens.size()-1).kind.equals("op")))) {
                int j = i + 1;
                StringBuilder numStr = new StringBuilder();
                numStr.append(ch);
                if (j < n && (Character.isDigit(expr.charAt(j)) ||
expr.charAt(j) == '.')) {
                    while (j < n && (Character.isDigit(expr.charAt(j))
|| expr.charAt(j) == '.')) {
                        numStr.append(expr.charAt(j));
                        j++;
                    }
                    tokens.add(new Token("num", numStr.toString()));
                    i = j;
                    continue;
                }
            }
            if (Character.isDigit(ch) || ch == '.') {
                int j = i;
                StringBuilder numStr = new StringBuilder();
                int dotCount = 0;
                while (j < n && (Character.isDigit(expr.charAt(j)) ||
expr.charAt(j) == '.')) {
                    if (expr.charAt(j) == '.') {
                        dotCount++;
                    }
                }
            }
        }
    }
}
```




<CODEWARS2026/>
LEÓN

```
        if (dotCount > 1) throw new
RuntimeException("ValueError");
    }
    numStr.append(expr.charAt(j));
    j++;
}
tokens.add(new Token("num", numStr.toString()));
i = j;
continue;
}
}
if (isOp(ch)) {
    tokens.add(new Token("op", String.valueOf(ch)));
    i++;
    continue;
}
throw new RuntimeException("ValueError");
}
if (tokens.isEmpty()) throw new RuntimeException("ValueError");
if (tokens.get(0).kind.equals("op") &&
!tokens.get(0).value.equals("+") && !tokens.get(0).value.equals("-"))
    throw new RuntimeException("ValueError");
if (tokens.get(tokens.size()-1).kind.equals("op"))
    throw new RuntimeException("ValueError");
for (int k = 0; k + 1 < tokens.size(); k++) {
    if (tokens.get(k).kind.equals("op") &&
tokens.get(k+1).kind.equals("op"))
        throw new RuntimeException("ValueError");
```



<CODEWARS2026/>
LEÓN

```
    }
    return tokens;
}

static boolean isOp(char ch) {
    return ch == '+' || ch == '-' || ch == '*' || ch == '/';
}

static List<Token> toRPN(List<Token> tokens) {
    Map<String,Integer> prec = new HashMap<>();
    prec.put("+", 2); prec.put("*", 2); prec.put("-", 1);
    prec.put("/", 1);
    List<Token> out = new ArrayList<>();
    Stack<Token> ops = new Stack<>();
    for (Token t : tokens) {
        if (t.kind.equals("num")) out.add(t);
        else {
            while (!ops.isEmpty() && prec.get(ops.peek().value) >=
prec.get(t.value)) {
                out.add(ops.pop());
            }
            ops.push(t);
        }
    }
    while (!ops.isEmpty()) out.add(ops.pop());
    return out;
}

static double evalRPN(List<Token> rpn) {
    Stack<Double> st = new Stack<>();
    for (Token t : rpn) {
```

<let's code/>





```
        if (t.kind.equals("num"))
st.push(Double.parseDouble(t.value));
        else {
            if (st.size() < 2) throw new
RuntimeException("ValueError");
            double b = st.pop();
            double a = st.pop();
            switch (t.value) {
                case "+": st.push(a + b); break;
                case "-": st.push(a - b); break;
                case "*": st.push(a * b); break;
                case "/":
                    if (b == 0) throw new
RuntimeException("ZeroDivisionError");
                    st.push(a / b); break;
                default: throw new RuntimeException("ValueError");
            }
        }
    }
    if (st.size() != 1) throw new RuntimeException("ValueError");
    return st.pop();
}

public static void main(String[] args) throws IOException {
    BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
    String expr = br.readLine();
    try {
        List<Token> tokens = tokenize(expr);
```



<CODEWARS2026/>
LEÓN

```
List<Token> rpn = toRPN(tokens);  
double res = evalRPN(rpn);  
if (Math.floor(res) == res) System.out.println((int)res);  
else System.out.println(res);  
} catch (RuntimeException e) {  
    System.out.println(e.getMessage());  
}  
}  
}
```

<let's code/>



20

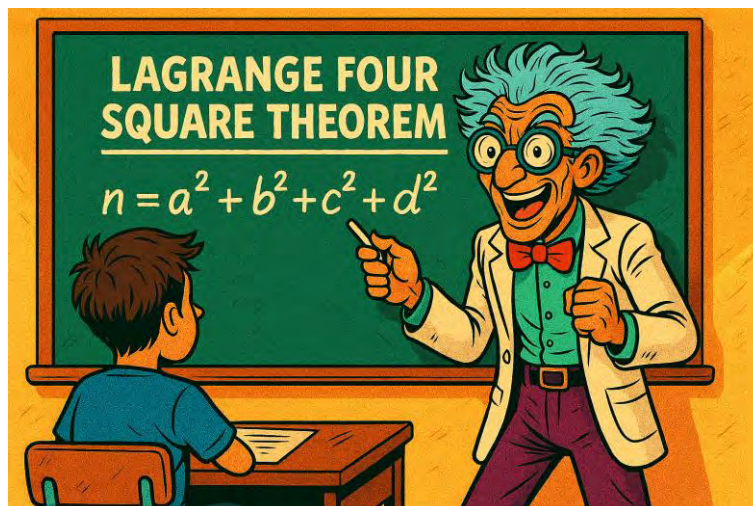
Lagrange's mouthshutter

12 points

Introduction

Recently, you had an argument with your mathematics teacher. He not only demands knowledge of all the theorems, but he has also become a devoted constructivist!

After you recited the Lagrange Four Square Theorem by heart, he now demands a computer program to obtain such a decomposition, to prove that you really understand the topic. How frustrating!



You remember the theorem well: Any positive integer can be decomposed as the sum of four squares of integers.

For example:

$$23 = 3^2 + 3^2 + 2^2 + 1^2$$

Or

$$1 = 1^2 + 0^2 + 0^2 + 0^2$$

In some cases, there are different alternatives:

$$59 = 7^2 + 3^2 + 1^2 + 0^2$$

$$59 = 5^2 + 5^2 + 3^2 + 0^2$$

$$59 = 5^2 + 4^2 + 3^2 + 3^2$$

That is:

$$N = a^2 + b^2 + c^2 + d^2$$

Exercise

To shut the teacher's mouth up, your task is to generate a program that returns the decomposition as 4 squares of a number (up to 1,000,000) passed as input, choosing the decomposition where the highest number is the largest first square. If a number has several alternatives with the same highest square, the option with the second highest square should be chosen and so on. For example, having as input the number 317 the ordered decompositions are:

$$317 = 16^2 + 6^2 + 5^2 + 0^2$$

$$317 = 16^2 + 6^2 + 4^2 + 3^2$$

$$317 = 14^2 + 11^2 + 0^2 + 0^2$$

...

Thus, the only correct answer is the first one because 16 is the highest first squared integer and, from those which have this same highest first squared integer, as they also have the same second squared integer, comparing the next value, 5 is the highest third squared integer.

A single solution will always be printed.

Input

A single number from 1 to 999999.

Output

The lowest value decomposition of the input number in squared integers following the Lagrange's Theorem.

As indicated, in some cases, there are different alternatives:

```
59 = 72 + 32 + 12 + 02  
59 = 52 + 52 + 32 + 02  
59 = 52 + 42 + 32 + 32  
...
```

When this happens, the appropriate one is that with the lowest "a", then the lowest "b", etc. Therefore a must be $\geq b$ and b must be $\geq c$ and c must be $\geq d$:

$$N = a^2 + b^2 + c^2 + d^2$$

Therefore in the previous example, the expected output is

$$59 = 5^2 + 4^2 + 3^2 + 3^2$$

Example 1

Input

```
1
```

Output

$$1 = 1^2 + 0^2 + 0^2 + 0^2$$

Example 2

Input

```
59
```

Output

$$59 = 5^2 + 4^2 + 3^2 + 3^2$$

Example 3

Input

23

Output

23 = 3^2 + 3^2 + 2^2 + 1^2

Solutions

Python

```
import math

def descomposition_lagrange_minmax(n: int):
    """
    Devuelve la descomposición  $n = a^2 + b^2 + c^2 + d^2$ 
    cumpliendo  $a \geq b \geq c \geq d \geq 0$ 
    y donde la tupla (a,b,c,d) es la lexicográficamente mínima.
    """
    # El mayor cuadrado ( $a^2$ ) debe cumplir  $a^2 \leq n$ .
    # Además, para 4 cuadrados, el valor mínimo posible de a es
    ceil(sqrt(n/4)).
    a_min = math.isqrt((n + 3) // 4)
    # Recorremos a desde su menor valor posible
    for a in range(a_min, math.isqrt(n) + 1):
        aa = a * a
        #  $b \leq a$ 
        b_max = min(a, math.isqrt(n - aa))
        # b de 0 ... b_max (minimiza b)
        for b in range(0, b_max + 1):
            bb = aa + b * b
```




<CODEWARS2026/>
LEÓN

```
        if bb > n:
            break
        #  $c \leq b$ 
        c_max = min(b, math.isqrt(n - bb))
        # c de 0 ... c_max (minimiza c)
        for c in range(0, c_max + 1):
            cc = bb + c * c
            if cc > n:
                break
            rest = n - cc
            d = math.isqrt(rest)
            #  $d \leq c$  y cuadrado perfecto
            if d <= c and d*d == rest:
                return (a, b, c, d)

    return None

# Uso
number = int(input().strip())
if 1 <= number <= 999999:
    sol = descomposition_lagrange_minmax(number)
    if sol:
        a,b,c,d = sol
        print(f"{number} = {a}^2 + {b}^2 + {c}^2 + {d}^2")
else:
    print("The number must be between 1 and 999999.")
```

C++

```
#include <bits/stdc++.h>
using namespace std;
// Raíz cuadrada entera segura (evita imprecisiones de casting)
```



<CODEWARS2026/>
LEÓN

```
static inline int isqrt_ll(long long x) {
    // sqrtl con long double es robusto y rápido para nuestro rango
    return static_cast<int>(sqrtl(static_cast<long double>(x)));
}

static inline int ceil_isqrt_div4(int n) {
    // ceil(sqrt(n/4)) -> la cota inferior del mayor sumando
    // equivalente entero: ceil( sqrt(n/4) ) = ceil( sqrt(n) / 2 )
    // Usamos una forma entera estable:
    // a_min = smallest a s.t. 4*a*a >= n
    int a = isqrt_ll((n + 3) / 4);          // aproximación por entero
    // Ajuste fino por si hiciera falta
    while (1LL * (a - 1) * (a - 1) * 4 >= n && a > 0) --a;
    while (1LL * a * a * 4 < n) ++a;
    return a;
}

optional<array<int,4>> lagrange_decomposition_minlex(int n) {
    // a es el mayor sumando; para minimizar (a,b,c,d) con a>=b>=c>=d,
    // empezamos en el menor a posible (ceil(sqrt(n/4))) y vamos subiendo.
    const int a_min = ceil_isqrt_div4(n);
    const int a_max = isqrt_ll(n);
    for (int a = a_min; a <= a_max; ++a) {
        long long aa = 1LL * a * a;
        // b <= a y b^2 <= n - a^2
        int b_upper = min(a, isqrt_ll(n - aa));
        for (int b = 0; b <= b_upper; ++b) {           // b crece ->
            minimiza b
            long long bb = aa + 1LL * b * b;
            if (bb > n) break;
        }
    }
}
```



<CODEWARS2026/>
LEÓN

```
// c <= b y c^2 <= n - a^2 - b^2
int c_upper = min(b, isqrt_ll(n - bb));
for (int c = 0; c <= c_upper; ++c) {           // c crece ->
minimiza c
    long long cc = bb + 1LL * c * c;
    if (cc > n) break;
    long long rest = n - cc;
    int d = isqrt_ll(rest);
    // exigir orden a>=b>=c>=d y que d^2 == rest
    if (d <= c && 1LL * d * d == rest) {
        return array<int,4>{a, b, c, d};      // primera
encontrada = lexicográficamente mínima
    }
}
}
}
return nullopt;
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    long long number;
    if (!(cin >> number)) return 0;
    if (1 <= number && number <= 999'999) {
        auto sol =
lagrange_decomposition_minlex(static_cast<int>(number));
        if (sol.has_value()) {
            auto v = *sol; // v[0]=a >= v[1]=b >= v[2]=c >= v[3]=d

```



<CODEWARS2026/>
LEÓN

```
        cout << number << " = "  
            << v[0] << "^2 + "  
            << v[1] << "^2 + "  
            << v[2] << "^2 + "  
            << v[3] << "^2\n";  
    }  
    } else {  
        cout << "The number must be between 1 and 999999.\n";  
    }  
    return 0;  
}
```

Java

```
import java.io.BufferedReader;  
import java.io.IOException;  
import java.io.InputStreamReader;  
import java.util.Arrays;  
public class lagrange {  
    // Raíz entera segura (evita errores de redondeo al castear)  
    private static int isqrt(long x) {  
        return (int) Math.sqrt((double) x);  
    }  
    private static int[] lagrangeDecomposition(int n) {  
        final int A = isqrt(n);  
        for (int a = 0; a <= A; a++) {  
            long aa = 1L * a * a;  
            for (int b = 0; b <= a; b++) { // b <= a  
                long bb = aa + 1L * b * b;  
                if (bb > n) break;  
            }  
        }  
    }  
}
```




<CODEWARS2026/>
LEÓN

```
        for (int c = 0; c <= b; c++) {                // c <= b
            long cc = bb + 1L * c * c;
            if (cc > n) break;
            long rest = n - cc;
            int d = isqrt(rest);
            // d <= c + exactitud
            if (d <= c && 1L * d * d == rest) {
                return new int[]{a, b, c, d};          // ya en orden
            }
        }
    }
}
return null;
}

public static void main(String[] args) throws IOException {
    BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
    String line = br.readLine();
    if (line == null) return;
    long number = Long.parseLong(line.trim());
    if (1 <= number && number <= 999_999) {
        int[] res = lagrangeDecomposition((int) number);
        if (res != null) {
            // Orden descendente, como haces con sorted(reverse=True)
            Arrays.sort(res);
            // Arrays.sort ascendente: invertimos al imprimir
            System.out.printf("%d = %d^2 + %d^2 + %d^2 + %d^2%n",
```



<CODEWARS2026/>
LEÓN

```
        number, res[3], res[2], res[1], res[0]);  
    }  
    } else {  
        System.out.println("The number must be between 1 and  
999999.");  
    }  
}  
}
```

<let's code/>



21

Supreme numbers

12 points

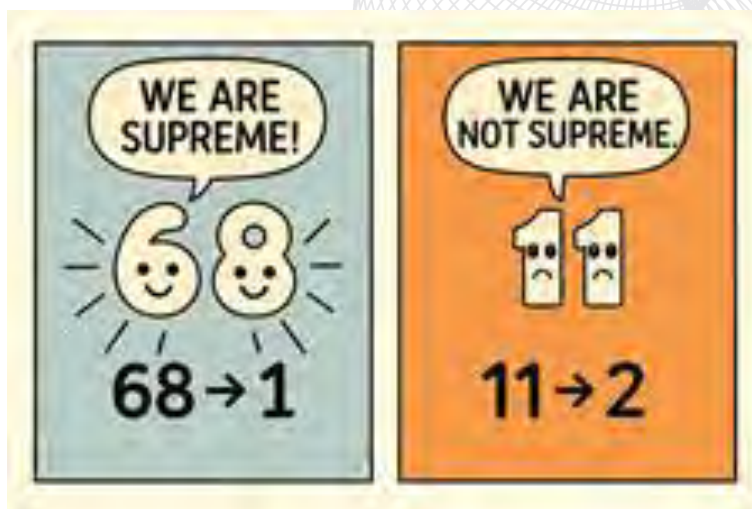
Introduction

Numeroland is the country of numbers. Numbers are generally happy, however inside numeroland inhabitants fall in different categories: There are supreme class numbers and there are normal class numbers.

Unfortunately not all numbers are supreme. For those of you who don't know, a supreme number is a number in which, if we repeatedly add the sum of the squares of its digits, at the end we get one (1). For example:

- 68 is a supreme number. why? Because $6^2 + 8^2 = 36 + 64 = 100$, and then $1^2 + 0^2 + 0^2 + 0^2 = 1$.
- 28 is a supreme number, because $2^2 + 8^2 = 4 + 64 = 68$. And then 68, as we have seen, is happy and in the end it adds up to 1.
- The 23 is a supreme number, because $2^2 + 3^2 = 4 + 9 = 13$. And then 13, is $1^2 + 3^2 = 10$ $1^2 + 0^2$ is 1.
- The number 11, on the other hand, is not supreme because $1^2 + 1^2 = 2$.

There are numbers that are supreme by birth: they alone are supreme (1, 28, 68, 100...). If they are grouped, they could stop being so (for instance $68 + 1 = 69$, as 69 is not supreme, both 68 and 1 stop being supreme. On the other hand non supreme number may become supreme when joining. Note that supreme numbers by birth are very selfing: they will not help to construct new supreme numbers. For instance with numbers 900 and 100 in the input, there will not be a chance to construct a new supreme number, because 100 is supreme by itself.



Exercise

Since we want all possible numbers to be supreme, you are asked to write a program that, given a series of positive numbers, obtain the groups of numbers that added together get as many supreme numbers as possible.

Input

The program will receive a list of numbers, finished by number "-1".

For example an input like:

```
34
50
50
-1
```

Note: The last -1 is to indicate where the list ends, it should not be taken into consideration. In this case all three numbers are initially not supreme.

Output

With the previous input, the program will indicate the numbers that are already 'supreme' and which unions of numbers make a new number supreme. The structure of the output will be:

```
Supreme numbers:  
<list of numbers already supreme>  
Unions that make supreme numbers:  
<list of griuping that make supreme numbers>
```

Therefore for this example the output would be:

```
Supreme numbers:  
None  
Unions that make supreme numbers:  
50+50
```

Note: The output should be like this, separated by the symbol "+".

Example 1

Input

```
100  
34  
50  
-1
```

Output

```
Supreme numbers:  
100  
Unions that make supreme numbers:  
None
```

Example 2

Input

```
100  
100  
50
```



<CODEWARS2026/>
LEÓN

```
50  
-1
```

Output

```
Supreme numbers:  
100, 100  
Unions that make supreme numbers:  
50+50
```

Example 3

Input

```
5  
5  
50  
50  
-1
```

Output

```
Supreme numbers:  
None  
Unions that make supreme numbers:  
5+5  
50+50
```

Solutions

Python

```
import itertools  
  
def es_feliz(num):  
    """Verifica si un número es feliz (supreme)."""  
    vistos = set()  
    while num != 1 and num not in vistos:  
        vistos.add(num)  
        num = sum(int(digit)**2 for digit in str(num))
```



<CODEWARS2026/> LEÓN

```
    return num == 1
def agrupar_supremos(numeros):
    """Agrupa los números en supremos, verificando combinaciones
    posibles."""
    supremos = []
    no_felices = []
    # Clasificar los números en felices y no felices
    for num in numeros:
        if es_feliz(num):
            supremos.append(num)
        else:
            no_felices.append(num)
    # Intentar encontrar combinaciones de números no felices que formen un
    número supreme
    uniones = []
    # Probar todas las combinaciones posibles de números no felices
    for r in range(2, len(no_felices) + 1): # Combinaciones de tamaño 2
        hasta el tamaño de la lista
        for combinacion in itertools.combinations(no_felices, r):
            # print(combinacion)
            suma = sum(combinacion)
            if es_feliz(suma):
                # Solo agregamos la combinación si el resultado no está ya
                en supremos
            # print(supremos)
            # if suma not in supremos:
            uniones.append(f"{'+'.join(map(str, combinacion))}")
    print(uniones)
```



<CODEWARS2026/>
LEÓN

```
#                 supremos.append(suma)
# Eliminar los números de la lista de no felices para
no volver a usarlos
    for num in combinacion:
        no_felices.remove(num)

print(uniones)
return supremos, uniones

def main():
    # Leemos la entrada
    numeros = []
    while True:
        num = int(input())
        if num == -1:
            break
        numeros.append(num)
    supremos, uniones = agrupar_supremos(numeros)
    # Mostrar los resultados
    print("Supreme numbers:")
    if supremos:
        print(", ".join(map(str, supremos)))
    else:
        print("None")
    print("Unions that make supreme numbers:")
    if uniones:
        for union in uniones:
            print(union)
    else:
        print("None")
```




<CODEWARS2026/> LEÓN

```
if __name__ == "__main__":  
    main()
```

C++

```
#include <iostream>  
#include <vector>  
#include <set>  
#include <sstream>  
#include <iterator>  
#include <algorithm>  
  
// Función para verificar si un número es feliz (supreme)  
bool esFeliz(int num) {  
    std::set<int> vistos;  
    while (num != 1 && vistos.find(num) == vistos.end()) {  
        vistos.insert(num);  
        int suma = 0;  
        while (num > 0) {  
            int digito = num % 10;  
            suma += digito * digito;  
            num /= 10;  
        }  
        num = suma;  
    }  
    return num == 1;  
}  
  
// Función para encontrar las uniones de números "supremes"  
void agruparSupremos(const std::vector<int>& numeros) {  
    std::vector<int> supremos;  
    std::vector<int> noFelices;
```



<CODEWARS2026/>
LEÓN

```
std::vector<std::string> uniones;
// Clasificar los números en felices (supremos) y no felices
for (int num : numeros) {
    if (esFeliz(num)) {
        supremos.push_back(num);
    } else {
        noFelices.push_back(num);
    }
}

// Intentar encontrar combinaciones de números no felices que sumen un
número "supreme"
for (size_t r = 2; r <= noFelices.size(); ++r) {
    for (size_t i = 0; i <= noFelices.size() - r; ++i) {
        int suma = 0;
        std::vector<int> combinacion;
        for (size_t j = 0; j < r; ++j) {
            combinacion.push_back(noFelices[i + j]);
            suma += noFelices[i + j];
        }
        if (esFeliz(suma)) {
            std::ostringstream oss;
            for (size_t j = 0; j < combinacion.size(); ++j) {
                oss << combinacion[j];
                if (j < combinacion.size() - 1) {
                    oss << "+";
                }
            }
            uniones.push_back(oss.str());
        }
    }
}
```



<CODEWARS2026/>
LEÓN

```
        }
    }
}
// Mostrar los resultados
if (!supremos.empty()) {
    std::cout << "Supreme numbers:\n";
    for (size_t i = 0; i < supremos.size(); ++i) {
        std::cout << supremos[i];
        if (i < supremos.size() - 1) {
            std::cout << ", ";
        }
    }
    std::cout << std::endl;
} else {
    std::cout << "Supreme numbers:\nNone\n";
}
if (!uniones.empty()) {
    std::cout << "Unions that make supreme numbers:\n";
    for (const std::string& unionStr : uniones) {
        std::cout << unionStr << std::endl;
    }
} else {
    std::cout << "Unions that make supreme numbers:\nNone\n";
}
}

int main() {
    std::vector<int> numeros;
    int num;
```

<let's code/>





<CODEWARS2026/> LEÓN

```
// Leer la entrada
while (std::cin >> num && num != -1) {
    numeros.push_back(num);
}
// Llamar a la función para agrupar los números supremes y las uniones
agruparSupremos(numeros);
return 0;
}
```

Java

```
import java.util.*;
import java.util.stream.Collectors;
public class NumeroSupreme {
    /** Verifica si un número es feliz (supreme). */
    public static boolean esFeliz(long num) {
        Set<Long> vistos = new HashSet<>();
        while (num != 1 && !vistos.contains(num)) {
            vistos.add(num);
            num = sumaCuadradosDigitos(num);
        }
        return num == 1;
    }
    private static long sumaCuadradosDigitos(long num) {
        long suma = 0;
        if (num < 0) num = -num; // por si llega algún negativo distinto
de -1
        while (num > 0) {
            long d = num % 10;
            suma += d * d;
        }
    }
}
```



```
        num /= 10;
    }
    return suma;
}

/** Contenedor de resultados: lista de supremos y lista de uniones. */
public static class Resultado {
    public final List<Long> supremos;
    public final List<String> uniones;
    public Resultado(List<Long> supremos, List<String> uniones) {
        this.supremos = supremos;
        this.uniones = uniones;
    }
}

/**
 * Agrupa los números en supremos, verificando combinaciones posibles
de no felices.
 * Replica la semántica del código Python:
 * - Clasifica en felices (supremos) y no felices.
 * - Prueba combinaciones de tamaño 2..N de los no felices (snapshot
por cada r).
 * - Si la suma de una combinación es feliz, se añade la unión y se
eliminan
 * dichos elementos de la lista de no felices para evitar
reutilización posterior.
 */
public static Resultado agruparSupremos(List<Long> numeros) {
    List<Long> supremos = new ArrayList<>();
    List<Long> noFelices = new ArrayList<>();
```

```
// Clasificar los números en felices y no felices
for (Long num : numeros) {
    if (esFeliz(num)) {
        supremos.add(num);
    } else {
        noFelices.add(num);
    }
}

// Intentar encontrar combinaciones de números no felices que
formen un número supreme
List<String> uniones = new ArrayList<>();
// Probar todas las combinaciones posibles de números no felices
(tamaño 2..len)
for (int r = 2; r <= noFelices.size(); r++) {
    // Generamos combinaciones a partir de un "snapshot" para
    replicar el comportamiento del Python
    List<List<Long>> combos = generarCombinaciones(new
    ArrayList<>(noFelices), r);
    for (List<Long> combinacion : combos) {
        long suma =
        combinacion.stream().mapToLong(Long::longValue).sum();
        if (esFeliz(suma)) {
            // Añadir unión en formato "a+b+..."
            String union = combinacion.stream()
                .map(String::valueOf)

.collect(Collectors.joining("+"));
            uniones.add(union);
        }
    }
}
```

```
// Eliminar números usados de noFelices (una
ocurrencia por valor)
for (Long val : combinacion) {
    noFelices.remove(val); // elimina la primera
coincidencia
}
}
}
}
return new Resultado(supremos, uniones);
}
/**
 * Genera todas las combinaciones de tamaño r a partir de la lista
base.
 * Implementación recursiva sin librerías externas.
 */
private static List<List<Long>> generarCombinaciones(List<Long> base,
int r) {
    List<List<Long>> resultado = new ArrayList<>();
    backtrackCombinaciones(base, r, 0, new ArrayList<>(), resultado);
    return resultado;
}
private static void backtrackCombinaciones(List<Long> base, int r, int
inicio,
                                List<Long> actual,
List<List<Long>> resultado) {
    if (actual.size() == r) {
        resultado.add(new ArrayList<>(actual));
    }
}
```

```
        return;
    }
    for (int i = inicio; i < base.size(); i++) {
        actual.add(base.get(i));
        backtrackCombinaciones(base, r, i + 1, actual, resultado);
        actual.remove(actual.size() - 1);
    }
}

/** Flujo principal: entrada por stdin, salida en español. */
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    List<Long> numeros = new ArrayList<>();
    // Lectura hasta -1
    while (true) {
        if (!sc.hasNextLong()) break; // fin de entrada
        long num = sc.nextLong();
        if (num == -1) break;
        numeros.add(num);
    }
    Resultado res = agruparSupremos(numeros);
    // Mostrar resultados
    System.out.println("Supreme numbers:");
    if (!res.supremos.isEmpty()) {
        System.out.println(res.supremos.stream()
                                .map(String::valueOf)
                                .collect(Collectors.joining(",
"")));
    } else {
```




<CODEWARS2026/>
LEÓN

```
        System.out.println("None");
    }
    System.out.println("Unions that make supreme numbers:");
    if (!res.uniones.isEmpty()) {
        for (String union : res.uniones) {
            System.out.println(union);
        }
    } else {
        System.out.println("None");
    }
}
}
```

<let's code/>



22

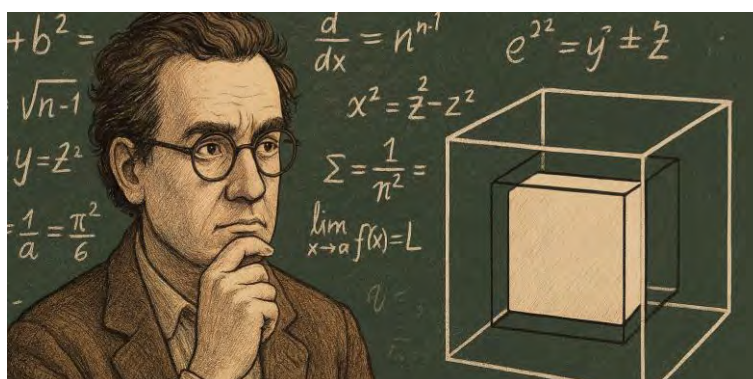
The Digits of the Perfect Square and Cube

12 points

Introduction

A perfect square is an integer that is the result of squaring another integer. That is, a number 'n' is a perfect square if there exists an integer k such that $n=k^2$. '9', '16' and '25' are perfect squares.

A perfect cube is an integer that is the result of raising another integer to the third power. That is, a number 'n' is a perfect cube if there exists an integer k such that $n=k^3$. '8', '27' and '64' are perfect cubes.



Exercise

You are asked to write a program that finds the smallest number that is both a perfect square and a perfect cube and that has (at least once) the digits indicated as input parameters. For example, if the numbers 1, 2, 4, and 6 are specified, you must find a number that is a perfect square and a perfect cube and has the digits 1, 2, 4, and 6 at least once.

The program will check up to the number 99999999. If no number is found, the program will write:

Could not find the specified Cube&Square number

Input

The program will receive an input like:

```
1;2;4;6
```

Note that inputted values are digits between 0 and 9, separated by ';'.

Output

...the output will be something like

<number>;<square root of number>;<cubic root of number>

For example:

```
262144;512;64
```

Note: This is because 262144 is 64^3 and at the same time 262144 is 512^2 . And at the same time 262144 is written with 1, 2, 4 and 6. Take into account, that you could have other digits different than 1,2,4 and 6, but AT LEAST these four need to be present once.

Example 1

Input

```
1;2;4;6
```

Output

```
262144;512;64
```

Example 2

Input

1;2;3

Output

11390625;3375;225

Example 3

Input

1

Output

1;1;1

Solutions

Python

```
#!/usr/bin/python3
def all_digits_in(digits, number):
    number_str = str(number)
    for digit in digits:
        if digit not in number_str:
            return False
    return True
def perfect_root(number, power):
    return pow(round(pow(number, 1/power)), power) == number
def test():
    print(perfect_root(262144, 2))
```




<CODEWARS2026/>
LEÓN

```
print(perfect_root(262144, 3))
print(all_digits_in(['1', '2', '4', '6'], 262144))
def main():
    digits = [i for i in input().split(';')]
    test_number = 0
    while True:
        if perfect_root(test_number, 3):
            if perfect_root(test_number, 2):
                if all_digits_in(digits, test_number):
                    print(str(test_number)+";"+str(round(pow(test_number,
1/2))))+";"+str(round(pow(test_number, 1/3))))
                    return
                test_number += 1
if __name__ == '__main__':
    main()
    #test()
```

C++

```
#include <iostream>
#include <cmath>
#include <vector>
#include <sstream>
bool allDigitsIn(const std::vector<std::string>& digits, long long number)
{
    std::string numberStr = std::to_string(number);
    for (const auto& digit : digits) {
        if (numberStr.find(digit) == std::string::npos) {
            return false;
        }
    }
}
```

<let's code/>





<CODEWARS2026/> LEÓN

```
    }
    return true;
}

bool perfectRoot(long long number, int power) {
    long long root = std::llround(std::pow(number, 1.0 / power));
    return std::pow(root, power) == number;
}

int main() {
    std::string line;
    std::getline(std::cin, line);
    std::vector<std::string> digits;
    std::stringstream ss(line);
    std::string item;
    while (std::getline(ss, item, ';')) {
        digits.push_back(item);
    }
    long long testNumber = 0;
    while (true) {
        if (perfectRoot(testNumber, 3)) {
            if (perfectRoot(testNumber, 2)) {
                if (allDigitsIn(digits, testNumber)) {
                    long long sqrt = std::llround(std::sqrt(testNumber));
                    long long cbrt = std::llround(std::pow(testNumber, 1.0
/ 3));

                    std::cout << testNumber << ";" << sqrt << ";" << cbrt
<< std::endl;

                    break;
                }
            }
        }
    }
}
```



<CODEWARS2026/>
LEÓN

```
        }  
    }  
    testNumber++;  
}  
return 0;  
}
```

Java

```
import java.util.Scanner;  
  
public class The_Digits_of_the_Perfect_Square_and_Cube {  
    public static boolean allDigitsIn(String[] digits, long number) {  
        String numberStr = Long.toString(number);  
        for (String digit : digits) {  
            if (!numberStr.contains(digit)) {  
                return false;  
            }  
        }  
        return true;  
    }  
  
    public static boolean perfectRoot(long number, int power) {  
        long root = Math.round(Math.pow(number, 1.0 / power));  
        return Math.pow(root, power) == number;  
    }  
  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        String[] digits = sc.nextLine().split(";");  
        long testNumber = 0;  
        while (true) {  
            if (perfectRoot(testNumber, 3)) {
```

<let's code/>





<CODEWARS2026/>
LEÓN

```
        if (perfectRoot(testNumber, 2)) {  
            if (allDigitsIn(digits, testNumber)) {  
                long sqrt = Math.round(Math.pow(testNumber, 0.5));  
                long cbrt = Math.round(Math.pow(testNumber, 1.0 /  
3));  
  
                System.out.println(testNumber + ";" + sqrt + ";" +  
cbrt);  
  
                break;  
            }  
        }  
        testNumber++;  
    }  
    sc.close();  
}  
}
```

<let's code/>



23

The knight's dilemma

12 points

Introduction

In the Kingdom of Argentvale, where the skies were pierced by towering castles and the lands rolled with emerald fields, there existed a peculiar legend. Deep within the royal archives lay a chessboard made of black onyx and ivory, said to be a gift from the gods themselves. Inscribed upon its edges were the words: *"Only the swift and the wise shall claim the Golden Chalice."* For generations, kings and queens failed to decipher its meaning, until the great scholar Magnus decoded the challenge: the knight's puzzle.



The knight, a gallant and unpredictable piece, was to traverse the board, starting from its southernmost corner, to reach a gleaming golden chalice resting at the northernmost square. However, the journey was fraught with peril. The path of the knight was governed by ancient rules: two steps forward, one step aside, in any direction—a symbol of the knight's unpredictable yet deliberate nature.

A Quest for Glory

The challenge drew many brave knights from across Argentvale. Among them was Sir Cedric of Ravenshollow, renowned for his cunning mind and valor. Aided by his loyal

squire, Edgar, Sir Cedric stood before the massive chessboard, its squares glowing faintly under the light of the full moon.

“Mark my words, Edgar,” Sir Cedric declared, his voice steady as his hand rested on the pommel of his sword, “this is no mere game. It is a test of strategy and persistence.”

The chalice shimmered in the moonlight, tantalizingly close yet so far. The board was enchanted; each wrong step would summon phantoms of fallen knights to haunt the challenger. With a deep breath, Sir Cedric mounted the board, each square radiating with ancient magic beneath his armored boots.

The First Move

The first steps were deliberate. Sir Cedric leaped two squares forward, then one to the side, just as the rules dictated. His moves echoed through the silent chamber, and Edgar hastily jotted down the path on parchment. “Two forward, one left,” murmured Edgar, his quill scratching furiously. “Remember, my lord, to think three steps ahead!”

As Sir Cedric advanced, the journey grew treacherous. Some squares were enchanted with illusions, showing the chalice closer than it truly was. Others whispered false guidance, tempting him to stray off course. Yet, Cedric pressed on, his mind sharp, recalling the wisdom of the knights before him: “The shortest path is not always the simplest.”

The Trial of Shadows

Halfway across the board, a ghostly knight rose from the onyx square Sir Cedric had just vacated. Its hollow eyes glowed with a pale blue fire, and its spectral blade gleamed. “Prove your worth, mortal,” the ghost intoned, blocking Cedric’s path.

With a flourish, Sir Cedric engaged the phantom. The battle was fierce but brief, for Cedric’s clever footwork mimicked the very moves he had made across the board. Each leap mirrored the knight’s unique motion, disorienting the specter until it dissolved into mist.

“Brilliant, my lord!” Edgar cheered, relief washing over him. “But the endgame approaches!”

The Final Leap

At last, Sir Cedric stood one move away from the chalice. The square before him glowed ominously—a trap, no doubt. Cedric studied the board, his mind racing. He recalled the ancient teachings of Magnus: “A knight’s journey is never direct, but always precise.”

Instead of moving forward, Sir Cedric took an unexpected route—leaping to a square off to the side, then doubling back toward the chalice in a sweeping arc. The board trembled as the final move landed. The golden chalice erupted in a shower of light, its brilliance filling the chamber.

Victory and Wisdom

Cedric lifted the chalice triumphantly. Its inscription gleamed in ancient runes: “The greatest strength lies not in force, but in foresight.” With a smile, he turned to Edgar. “Remember this day, my friend. Strategy and patience have won the hour.”

The tale of Sir Cedric spread across Argentvale, inspiring knights, scholars, and dreamers alike. The knight’s puzzle became a symbol of wisdom and determination, a lesson that echoed through the ages: every leap, though unconventional, brings one closer to destiny.

Exercise

Write a program that calculates the movements of a knight from its initial position to the final position, indicating the path and the number of movements needed.

Input

The program will receive three lines:



<CODEWARS2026/> LEÓN

- The Size of the chess table
- The Initial position of the knight (x y)
- The Desired final position of the knight(x y)

For instance:

```
8
1 1
7 7
```

Output

The program will output two lines. The first one is the number of movements needed. The second line is the path followed. Starting from the initial position and ending in the final position, the line will show #NumberOfMovements-1 more positions showing the path from the beginning to the end.

It will show something like:

```
4
(1, 1) (2, 3) (3, 5) (5, 6) (7, 7)
```

Example 1

Input

```
8
1 1
7 7
```

Output

```
4
(1, 1) (2, 3) (3, 5) (5, 6) (7, 7)
```

Example 2

Input



<CODEWARS2026/>
LEÓN

```
12
5 5
9 12
```

Output

```
5
(5, 5) (7, 4) (6, 6) (7, 8) (8, 10) (9, 12)
```

Solutions

Python

```
from collections import deque
# Utility function to check if (x, y) lies inside the board
def is_inside(x, y, N):
    return 1 <= x <= N and 1 <= y <= N
# Method to calculate the minimum steps and the path to reach the target
position
def min_steps_to_reach_target(knight_pos, target_pos, N):
    # x and y direction, where a knight can move
    dx = [-2, -1, 1, 2, -2, -1, 1, 2]
    dy = [-1, -2, -2, -1, 1, 2, 2, 1]
    # Queue for storing states of the knight on the board
    q = deque()
    # Push the starting position of the knight with 0 distance and initial
path
    initial_path = [(knight_pos[0], knight_pos[1])]
    q.append((knight_pos[0], knight_pos[1], 0, initial_path))
    # Visited set to keep track of visited cells
    visited = set()
    visited.add((knight_pos[0], knight_pos[1]))
    while q:
```



<CODEWARS2026/>
LEÓN

```
x, y, dist, path = q.popleft()
# If the current cell is the target cell, print the path and
return its distance
if (x, y) == (target_pos[0], target_pos[1]):
    print(dist)
    for step in path:
        print(f"({step[0]}, {step[1]})", end=" ")
    print()
    return dist
# Loop for all reachable states
for i in range(8):
    new_x, new_y = x + dx[i], y + dy[i]
    # If reachable state is not yet visited and inside the board,
    push that state into the queue
    if is_inside(new_x, new_y, N) and (new_x, new_y) not in
    visited:
        visited.add((new_x, new_y))
        new_path = path + [(new_x, new_y)]
        q.append((new_x, new_y, dist + 1, new_path))
# If no valid path is found, return -1
print("No valid path found to reach the target position.")
return -1
# Driver code
if __name__ == "__main__":
    N = int(input()) # Board size
    knight_pos = tuple(map(int, input().split())) # Starting position of
the knight
    target_pos = tuple(map(int, input().split())) # Target position
```



```
    if not is_inside(knight_pos[0], knight_pos[1], N) or not  
is_inside(target_pos[0], target_pos[1], N):  
        print("Invalid positions! Both positions must be within the  
chessboard.")  
    else:  
        min_steps_to_reach_target(knight_pos, target_pos, N)
```

C++

```
#include <bits/stdc++.h>  
using namespace std;  
// Structure for storing a cell's data  
struct cell {  
    int x, y, dis;  
    vector<pair<int, int>> path; // To store the path  
    cell(int x, int y, int dis, vector<pair<int, int>> path)  
        : x(x), y(y), dis(dis), path(path) {}  
};  
// Utility method to check if (x, y) lies inside the board  
bool isInside(int x, int y, int N) {  
    return x >= 1 && x <= N && y >= 1 && y <= N;  
}  
// Method to calculate the minimum steps and the path to reach the target  
position  
int minStepToReachTarget(int knightPos[], int targetPos[], int N) {  
    // x and y direction, where a knight can move  
    int dx[] = { -2, -1, 1, 2, -2, -1, 1, 2 };  
    int dy[] = { -1, -2, -2, -1, 1, 2, 2, 1 };  
    // Queue for storing states of the knight on the board  
    queue<cell> q;
```



<CODEWARS2026/>
LEÓN

```
// Push the starting position of the knight with 0 distance and
initial path
vector<pair<int, int>> initialPath = { {knightPos[0], knightPos[1]} };
q.push(cell(knightPos[0], knightPos[1], 0, initialPath));
// Visited array to keep track of visited cells
vector<vector<bool>> visit(N + 1, vector<bool>(N + 1, false));
visit[knightPos[0]][knightPos[1]] = true;
while (!q.empty()) {
    cell t = q.front();
    q.pop();
    // If the current cell is the target cell, print the path and
return its distance
    if (t.x == targetPos[0] && t.y == targetPos[1]) {
        //cout << "Minimum steps to reach the target position: " <<
t.dis << endl;
        cout << t.dis << endl;
        //cout << "Path:" << endl;
        for (auto step : t.path) {
            cout << "(" << step.first << ", " << step.second << ") ";
        }
        cout << endl;
        return t.dis;
    }
    // Loop for all reachable states
    for (int i = 0; i < 8; i++) {
        int x = t.x + dx[i];
        int y = t.y + dy[i];
```




<CODEWARS2026/>
LEÓN

```
        // If reachable state is not yet visited and inside the board,
        push that state into the queue
        if (isInside(x, y, N) && !visit[x][y]) {
            visit[x][y] = true;
            vector<pair<int, int>> newPath = t.path;
            newPath.push_back({ x, y });
            q.push(cell(x, y, t.dis + 1, newPath));
        }
    }
}

// If no valid path is found, return -1
cout << "No valid path found to reach the target position." << endl;
return -1;
}

// Driver code
int main() {
    int N;
    // Input the board size
    //cout << "Enter the size of the chessboard: ";
    cin >> N;
    int knightPos[2], targetPos[2];
    // Input the knight's starting position
    //cout << "Enter the starting position of the knight (x y): ";
    cin >> knightPos[0] >> knightPos[1];
    // Input the target position
    //cout << "Enter the target position (x y): ";
    cin >> targetPos[0] >> targetPos[1];
    // Check for valid positions
```

```
    if (!isInside(knightPos[0], knightPos[1], N) ||  
        !isInside(targetPos[0], targetPos[1], N)) {  
        cout << "Invalid positions! Both positions must be within the  
chessboard." << endl;  
        return 0;  
    }  
    // Function call  
    minStepToReachTarget(knightPos, targetPos, N);  
    return 0;  
}
```

Java

```
import java.util.*;  
class knight {  
    static class Cell {  
        int x, y, dist;  
        List<int[]> path;  
        Cell(int x, int y, int dist, List<int[]> path) {  
            this.x = x;  
            this.y = y;  
            this.dist = dist;  
            this.path = path;  
        }  
    }  
    // Utility method to check if (x, y) lies inside the board  
    static boolean isInside(int x, int y, int N) {  
        return x >= 1 && x <= N && y >= 1 && y <= N;  
    }  
}
```

```
// Method to calculate the minimum steps and the path to reach the
target position
static int minStepsToReachTarget(int[] knightPos, int[] targetPos, int
N) {
    // x and y direction, where a knight can move
    int[] dx = { -2, -1, 1, 2, -2, -1, 1, 2 };
    int[] dy = { -1, -2, -2, -1, 1, 2, 2, 1 };
    // Queue for storing states of the knight on the board
    Queue<Cell> q = new LinkedList<>();
    // Push the starting position of the knight with 0 distance and
initial path
    List<int[]> initialPath = new ArrayList<>();
    initialPath.add(new int[] { knightPos[0], knightPos[1] });
    q.add(new Cell(knightPos[0], knightPos[1], 0, initialPath));
    // Visited set to keep track of visited cells
    boolean[][] visited = new boolean[N + 1][N + 1];
    visited[knightPos[0]][knightPos[1]] = true;
    while (!q.isEmpty()) {
        Cell t = q.poll();
        // If the current cell is the target cell, print the path and
return its distance
        if (t.x == targetPos[0] && t.y == targetPos[1]) {
            System.out.println(t.dist);
            for (int[] step : t.path) {
                System.out.print("(" + step[0] + ", " + step[1] + ")
");
            }
            System.out.println();
        }
    }
}
```



<CODEWARS2026/>
LEÓN

```
        return t.dist;
    }
    // Loop for all reachable states
    for (int i = 0; i < 8; i++) {
        int x = t.x + dx[i];
        int y = t.y + dy[i];
        // If reachable state is not yet visited and inside the
board, push that state into the queue
        if (isInside(x, y, N) && !visited[x][y]) {
            visited[x][y] = true;
            List<int[]> newPath = new ArrayList<>(t.path);
            newPath.add(new int[] { x, y });
            q.add(new Cell(x, y, t.dist + 1, newPath));
        }
    }
}
// If no valid path is found, return -1
System.out.println("No valid path found to reach the target
position.");
return -1;
}
// Driver code
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int N = sc.nextInt(); // Board size
    int[] knightPos = { sc.nextInt(), sc.nextInt() }; // Starting
position of the knight
```

<let's code/>





<CODEWARS2026/>
LEÓN

```
int[] targetPos = { sc.nextInt(), sc.nextInt() }; // Target
position
    if (!isInside(knightPos[0], knightPos[1], N) ||
        !isInside(targetPos[0], targetPos[1], N)) {
        System.out.println("Invalid positions! Both positions must be
within the chessboard.");
    } else {
        minStepsToReachTarget(knightPos, targetPos, N);
    }
    sc.close();
}
```

<let's code/>



24

The Enigmatic multiplication

14 points

Introduction

Sherlock Holmes is investigating an strange case and has found many papers with Enigmatic Multiplications in them. He soon discovers that in them the numbers have been replaced by letters. He has called you as he knows you are one of those rare programmers that exist in the 23IX century...



Exercise

You are asked to develop a program capable of discovering a multiplication in which the numbers have been replaced by letters. It will be a multiplication like the following:

```

  1 2
x 3 4
---
 4 8
3 6
----
4 0 8

```

but with letters:

```

A B

```

```
C D
---
D G
C H
-----
D F G
```

Your program will receive a two-operand multiplication and return the 'lowest' solution.

Input

The program will receive as input a series of lines corresponding to the multiplicands, the partial results, and the final result. The input will end with a semicolon (";"). The digits will be separated among them with an space.

To explain the location of each digit, in this example the left spaces will be filled with dots. For instance the previous multiplication's input will be inputted like:

```
. A B
. C D
. D G
C H
D F G
;
```

Note 1: This exercise will ALWAYS receive a 2 number multiplication (but each number could have 2, 3 or more 2 digits, note it), therefore the two first lines are the numbers that will be multiplied. The number of intermediate lines depend on the number of digits of the numbers multiplied.

Note 2: Separator lines ("---") are not introduced. They appear in examples only for clarity.

Output

The program will show the output with the following format:

```
A=1;B=2;C=3;D=4;F=0;G=8;H=6
```

It should be ordered by letter

Note 1: The multiplicands will NOT start by 0

Note 2: To avoid that an input may generate different solutions, you are asked to show only the 'lowest' solution. That is, the solution where the A is lowest, B is lowest, C is lowest... For instance if solutions may be

```
A=1;B=2;C=1  
A=1;B=3;C=0
```

You will show the first line: That with A being lowest, in case of equality, that with B being lowest, etc.

Example 1

Input

```
. A B  
. C D  
. D G  
C H  
D F G  
;
```

Output

```
A=1;B=2;C=3;D=4;F=0;G=8;H=6
```

Example 2

Input

```
. . A B C  
. . . D A  
. E E F G  
A F A H  
A C G H G  
;
```

Output

A=7;B=8;C=6;D=9;E=1;F=0;G=2;H=4

Solutions

Python

```
import itertools

def count_leading_dots(tokens):
    """Cuenta los puntos ('.') a la izquierda hasta el primer token no
    punto."""
    c = 0
    for t in tokens:
        if t == '.':
            c += 1
        else:
            break
    return c

def parse_input():
    lines = []
    while True:
        line = input().rstrip()
        if line == ";":
            break
        if line: # ignorar líneas vacías
            lines.append(line)
    return lines

def extract_letters(lines):
    letters = set()
    for line in lines:
        for ch in line:
```

```
        if ch.isalpha():
            letters.add(ch)
    return sorted(letters)

def tokens_from_line(line):
    """Devuelve la lista de tokens separados por espacios (cada token será
    '.' o una letra)."""
    return line.split()

def build_number_from_tokens(tokens, mapping):
    """Construye la cadena de dígitos (sin puntos) a partir de tokens
    usando mapping."""
    s = []
    for t in tokens:
        if t == '.':
            continue
        # t debe ser una letra (según la especificación)
        if t.isalpha():
            s.append(mapping[t])
        else:
            # por seguridad, si hubiera dígito directo
            s.append(t)
    return ''.join(s)

def find_leading_letters(lines):
    """
    Devuelve el conjunto de letras que son la primera (más a la izquierda)
    cifra de algún número que contenga más de una cifra.
    """
    leading = set()
    for line in lines:
```



<CODEWARS2026/>
LEÓN

```
tokens = tokens_from_line(line)
# contar tokens relevantes (no '.')
relevant = [t for t in tokens if t != '.']
if len(relevant) <= 1:
    continue
# buscar la primera token relevante; si es letra, anotarla
for t in tokens:
    if t == '.':
        continue
    if t.isalpha():
        leading.add(t)
    # si es dígito literal (no esperado), no añadimos
    break
return leading
def solve(lines):
    """
    En letters se saca el listado de letras ordenado, A,B,C,D... sin
    ningun orden
    """
    letters = extract_letters(lines)
    if len(letters) > 10:
        return None # imposible: más de 10 letras
    # tokens por línea para reutilizar
    """
    Genera una lista de listas como: [['.', 'A', 'B'], ['.', 'C', 'D'],
    ...]
    """
    lines_tokens = [tokens_from_line(line) for line in lines]
```



<CODEWARS2026/> LEÓN

```
# conjunto de letras que no pueden valer 0 (leading letters of multi-
digit numbers)
leading_letters = find_leading_letters(lines)
# identificamos multiplicando(s) y resultado final:
# según el enunciado las dos primeras líneas son los factores y la
última es el resultado final
# (las demás son parciales). Nos ceñimos a verificar multiplicand1 *
multiplicand2 == final_result.
# Si deseas comprobar parciales, podemos añadirlo.
# Convertiremos cada línea en número (ignorando líneas vacías).
for perm in itertools.permutations('0123456789', len(letters)):
    mapping = dict(zip(letters, perm))
    # poda: si alguna letra líder se mapea a '0', saltar
    if any(mapping[l] == '0' for l in leading_letters):
        continue
    try:
        numbers = []
        for tokens in lines_tokens:
            #tokens es ['.', 'A', 'B'] y similar y numstr es un
posible numero que podria encajar con .AB
            numstr = build_number_from_tokens(tokens, mapping)
            if numstr == '':
                # línea sin números (posible), la ignoramos
                continue
            # evitar números con ceros a la izquierda (si la
representación tiene >1 dígito y empieza con '0')
            if len(numstr) > 1 and numstr[0] == '0':
                raise ValueError("leading zero")
```



```
        numbers.append(int(numstr))
    if len(numbers) < 3:
        continue
    multiplicand1 = numbers[0]
    multiplicand2 = numbers[1]
    final_result = numbers[-1]
    if multiplicand1 * multiplicand2 != final_result:
        continue
    #print(mapping,"--",tokens,"--",numstr)
    # ===== Verificación de productos parciales con alineado por
puntos =====
    # Asumimos:
    #   - total_columns = número de tokens de la línea del
resultado (todas deberían tener el mismo ancho).
    #   - los parciales son lines_tokens[2:-1], en orden: primero
el parcial de las unidades.
    total_columns = len(lines_tokens[-1])
    partial_lines = lines_tokens[2:-1]
    # reconstruimos la cadena del segundo factor para inspeccionar
sus dígitos de derecha a izquierda
    m2_str = build_number_from_tokens(lines_tokens[1], mapping)
    if m2_str == '':
        # segundo factor vacío → inválido
        continue
    # Si hay líneas parciales presentadas, deben corresponder a
los dígitos de m2.
    # Permitimos que haya el mismo número de parciales que dígitos
de m2.
```

```
# (Si tu formato permite omitir parciales que salen 0, este
bloque ya lo contempla más abajo.)
if len(partial_lines) != len(m2_str):
    # Si quieres permitir discrepancias, puedes relajar esta
    restricción,
    # pero por defecto exigimos mismo número de parciales que
    dígitos en multiplicand2.
    continue
# Verificamos cada parcial i (0 = unidades)
ok_partials = True
for i, partial_tokens in enumerate(partial_lines):
    digit = int(m2_str[-1 - i]) # dígito i desde la derecha
    expected_value_str = str(multiplicand1 * digit)
    # expected_left_dots = columnas totales - (longitud del
    valor sin shift + shift i)
    expected_left_dots = total_columns -
    (len(expected_value_str) + i)
    # conteo real de padding
    actual_left_dots = count_leading_dots(partial_tokens)
    actual_numstr = build_number_from_tokens(partial_tokens,
mapping)

    # Validaciones:
    # 1) padding a la izquierda debe coincidir
    # - salvo el caso especial digit==0 con línea vacía:
    permitimos que actual_numstr == ''
    # y entonces solo exigimos que la línea sean todo
    puntos (padding completo).
    if digit == 0:
```

```
# Aceptamos:
#   a) línea completamente vacía (solo puntos) ->
actual_numstr == ''

#   b) línea con "0" alineado exactamente ->
actual_numstr == '0' y padding coincide
only_dots = all(t == '.' for t in partial_tokens)
if only_dots:
    # Para "solo puntos", aceptamos como válido
    # (representa 0 desplazado)
    continue
else:
    # Debe ser un "0" exactamente y padding exacto
    if not (actual_numstr == '0' and actual_left_dots
== expected_left_dots):

        ok_partials = False
        break

else:
    # Para digit != 0 debe coincidir valor y padding
    if not (actual_numstr == expected_value_str and
actual_left_dots == expected_left_dots):
        ok_partials = False
        break

if not ok_partials:
    continue

# ===== Fin verificación de productos parciales =====
return mapping
except ValueError:
    continue
```



<CODEWARS2026/> LEÓN

```
    return None

def main():
    lines = parse_input()
    mapping = solve(lines)
    if mapping:
        # imprimir en orden alfabético sin espacio final: A=6;B=3;...
        out = ';'.join(f"{k}={mapping[k]}" for k in sorted(mapping))
        print(out)
    else:
        print("No solution found")

if __name__ == "__main__":
    main()
```

C++

```
#include <bits/stdc++.h>
using namespace std;
// === Utilidades de tokenización y construcción ===
static vector<string> tokens_from_line(const string& line) {
    vector<string> tokens;
    istringstream iss(line);
    string tok;
    while (iss >> tok) tokens.push_back(tok);
    return tokens;
}

static int count_leading_dots(const vector<string>& tokens) {
    int c = 0;
    for (const auto& t : tokens) {
        if (t == ".") ++c;
        else break;
    }
}
```




<CODEWARS2026/>
LEÓN

```
    }  
    return c;  
}  
static string build_number_from_tokens(const vector<string>& tokens,  
                                       const unordered_map<char, char>&  
mapping) {  
    string s;  
    for (const auto& t : tokens) {  
        if (t == ".") continue;  
        // Por especificación: t será '.' o una letra; aun así,  
        // si llega un dígito literal, lo copiamos por seguridad.  
        if (t.size() == 1 && isalpha(static_cast<unsigned char>(t[0]))) {  
            auto it = mapping.find(t[0]);  
            if (it == mapping.end()) {  
                // No debería ocurrir cuando el mapeo está completo;  
                // devolvemos vacío para indicar invalidez.  
                return "";  
            }  
            s.push_back(it->second);  
        } else {  
            // Copiamos token "tal cual" (podría ser un dígito literal)  
            s += t;  
        }  
    }  
    return s;  
}  
static vector<string> parse_input() {  
    vector<string> lines;
```

```
string line;
while (true) {
    if (!std::getline(cin, line)) break;
    // elimina trailing spaces
    while (!line.empty() && (line.back() == '\\r' || line.back() == ' '
|| line.back() == '\\t'))
        line.pop_back();
    if (line == ";") break;
    if (!line.empty()) lines.push_back(line); // ignorar líneas vacías
}
return lines;
}

static vector<char> extract_letters(const vector<string>& lines) {
    set<char> letters_set;
    for (const auto& line : lines) {
        for (unsigned char ch : line) {
            if (isalpha(ch)) letters_set.insert(static_cast<char>(ch));
        }
    }
    vector<char> letters(letters_set.begin(), letters_set.end());
    return letters;
}

static set<char> find_leading_letters(const vector<string>& lines) {
    set<char> leading;
    for (const auto& line : lines) {
        auto tokens = tokens_from_line(line);
        vector<string> relevant;
        for (const auto& t : tokens) if (t != ".") relevant.push_back(t);
    }
}
```

```
        if (relevant.size() <= 1) continue;
        for (const auto& t : tokens) {
            if (t == ".") continue;
            if (t.size() == 1 && isalpha(static_cast<unsigned
char>(t[0]))) {
                leading.insert(t[0]);
            }
            // si es dígito literal, no añadimos
            break;
        }
    }
    return leading;
}

// === Comprobación completa de un mapeo ===
static bool check_mapping(const unordered_map<char, char>& mapping,
                        const vector<vector<string>>& lines_tokens) {
    // Construye números de cada línea (ignorando las vacías)
    vector<long long> numbers;
    vector<string> numstr_per_line;
    numstr_per_line.reserve(lines_tokens.size());
    for (const auto& tokens : lines_tokens) {
        string numstr = build_number_from_tokens(tokens, mapping);
        numstr_per_line.push_back(numstr);
        if (numstr.empty()) {
            // línea sin números (posible), la ignoramos en 'numbers'
            continue;
        }
        if (numstr.size() > 1 && numstr[0] == '0') {
```



<CODEWARS2026/>
LEÓN

```
        // cero a la izquierda
        return false;
    }
    // Conversión a entero; asume que cabe en 64 bits
    // (si los casos reales exceden 18-19 dígitos, habría que usar
    enteros grandes por string).
    long long val = 0;
    for (char c : numstr) {
        if (!isdigit(static_cast<unsigned char>(c))) return false;
        int d = c - '0';
        // Chequeo simple de overflow (conservador)
        if (val > (LLONG_MAX - d) / 10) return false;
        val = val * 10 + d;
    }
    numbers.push_back(val);
}
if (numbers.size() < 3) return false;
long long multiplicand1 = numbers[0];
long long multiplicand2 = numbers[1];
long long final_result = numbers.back();
if (multiplicand1 * multiplicand2 != final_result) return false;
// === Verificación de productos parciales con alineado por puntos ===
int total_columns = static_cast<int>(lines_tokens.back().size());
if (total_columns <= 0) return false;
// parciales: lines_tokens[2 .. n-2]
vector<vector<string>> partial_lines;
if (lines_tokens.size() >= 4) { // al menos 2 factores + 1 parcial +
resultado
```



```
        for (size_t i = 2; i + 1 < lines_tokens.size(); ++i) {
            partial_lines.push_back(lines_tokens[i]);
        }
    }

    // reconstruir cadena del segundo factor (por tokens) para
    inspeccionar dígitos
    string m2_str = build_number_from_tokens(lines_tokens[1], mapping);
    if (m2_str.empty()) return false;
    // Debe haber tantos parciales como dígitos del segundo factor
    if (static_cast<int>(partial_lines.size()) !=
        static_cast<int>(m2_str.size()))
        return false;

    // Verificar cada parcial i (0 = unidades)
    for (int i = 0; i < static_cast<int>(partial_lines.size()); ++i) {
        const auto& partial_tokens = partial_lines[i];
        int digit = m2_str[static_cast<int>(m2_str.size()) - 1 - i] - '0';
        long long expected_val = multiplicand1 * digit;
        string expected_value_str = to_string(expected_val);
        int expected_left_dots = total_columns -
            (static_cast<int>(expected_value_str.size()) + i);
        int actual_left_dots = count_leading_dots(partial_tokens);
        string actual_numstr = build_number_from_tokens(partial_tokens,
            mapping);
        if (digit == 0) {
            // Aceptamos: a) solo puntos; b) "0" alineado
            bool only_dots = true;
            for (const auto& t : partial_tokens) {
                if (t != ".") { only_dots = false; break; }
            }
        }
    }
}
```



<CODEWARS2026/>
LEÓN

```
    }
    if (only_dots) {
        continue; // válido como parcial 0 desplazado
    } else {
        if (!(actual_numstr == "0" && actual_left_dots ==
expected_left_dots)) {
            return false;
        }
    }
} else {
    if (!(actual_numstr == expected_value_str && actual_left_dots
== expected_left_dots)) {
        return false;
    }
}
return true;
}

// === Búsqueda por backtracking: asignar letras -> dígitos ===
struct Solver {
    const vector<string> lines;
    const vector<vector<string>> lines_tokens;
    const vector<char> letters;           // letras ordenadas
    const set<char> leading_letters;     // no pueden ser '0'
    // Estado de backtracking
    unordered_map<char, char> mapping;   // letra -> dígito (char
'0'..'9')
    array<bool, 10> used{};              // dígitos usados
};
```



<CODEWARS2026/>
LEÓN

```
Solver(vector<string> lines_)
: lines(lines_),
  lines_tokens([&](){
    vector<vector<string>> v; v.reserve(lines_.size());
    for (const auto& ln : lines_)
      v.push_back(tokens_from_line(ln));
    return v;
  }()),
  letters(extract_letters(lines_)),
  leading_letters(find_leading_letters(lines_)) {
  used.fill(false);
}

bool solve_all_assigned() {
  // Validar mapeo completo
  return check_mapping(mapping, lines_tokens);
}

bool dfs_assign(int idx) {
  if (idx == static_cast<int>(letters.size())) {
    return solve_all_assigned();
  }
  char L = letters[idx];
  for (int d = 0; d <= 9; ++d) {
    if (used[d]) continue;
    if (d == 0 && leading_letters.count(L)) continue; // poda:
    líderes no pueden ser 0
    mapping[L] = static_cast<char>('0' + d);
    used[d] = true;
    // (Opcional: se puede añadir poda adicional aquí)
```



<CODEWARS2026/>
LEÓN

```
        if (dfs_assign(idx + 1)) return true;
        used[d] = false;
        mapping.erase(L);
    }
    return false;
}

// Punto de entrada
bool run(unordered_map<char, char>& out_mapping) {
    if (letters.size() > 10) return false; // imposible
    bool ok = dfs_assign(0);
    if (ok) out_mapping = mapping;
    return ok;
}

};

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    vector<string> lines = parse_input();
    Solver solver(lines);
    unordered_map<char, char> mapping;
    if (solver.run(mapping)) {
        // imprimir en orden alfabético sin espacio final: A=6;B=3;...
        vector<char> keys = solver.letters; // ya vienen ordenadas
        string out;
        for (size_t i = 0; i < keys.size(); ++i) {
            char k = keys[i];
            out += k;
            out += '=';
        }
    }
}
```




<CODEWARS2026/>
LEÓN

```
        out += mapping[k];
        if (i + 1 < keys.size()) out += ' ';
    }
    cout << out << "\n";
} else {
    cout << "No solution found\n";
}
return 0;
}
```

Java

```
import java.io.*;
import java.math.BigInteger;
import java.util.*;
import java.util.stream.Collectors;

public class The_Enigmatic_Multiplication {
    // === Parseo y utilidades de tokens ===
    static List<String> parseInput(BufferedReader br) throws IOException {
        List<String> lines = new ArrayList<>();
        String line;
        while ((line = br.readLine()) != null) {
            line = rstrip(line);
            if (line.equals(";")) break;
            if (!line.isEmpty()) lines.add(line); // ignorar líneas vacías
        }
        return lines;
    }

    static String rstrip(String s) {
        int i = s.length() - 1;
```

<let's code/>





<CODEWARS2026/>
LEÓN

```
        while (i >= 0 && (s.charAt(i) == '\r' || s.charAt(i) == ' ' ||  
s.charAt(i) == '\t')) i--;  
        return s.substring(0, i + 1);  
    }  
    static List<String> tokensFromLine(String line) {  
        if (line.isEmpty()) return Collections.emptyList();  
        String[] parts = line.trim().split("\\s+");  
        return Arrays.asList(parts);  
    }  
    static int countLeadingDots(List<String> tokens) {  
        int c = 0;  
        for (String t : tokens) {  
            if (t.equals(".")) c++;  
            else break;  
        }  
        return c;  
    }  
    static String buildNumberFromTokens(List<String> tokens,  
Map<Character, Character> mapping) {  
        StringBuilder sb = new StringBuilder();  
        for (String t : tokens) {  
            if (t.equals(".")) continue;  
            if (t.length() == 1 && Character.isAlphabetic(t.charAt(0))) {  
                Character mapped = mapping.get(t.charAt(0));  
                if (mapped == null) {  
                    // No debería pasar cuando mapping está completo;  
devolvemos vacío como inválido  
                    return "";  
                }  
            }  
            sb.append(mapped);  
        }  
        return sb.toString();  
    }  
}
```

<let's code/>





<CODEWARS2026/>
LEÓN

```
        }
        sb.append(mapped);
    } else {
        // Copiamos el token tal cual (si fuera dígito literal)
        sb.append(t);
    }
}
return sb.toString();
}

static List<Character> extractLetters(List<String> lines) {
    SortedSet<Character> set = new TreeSet<>();
    for (String line : lines) {
        for (int i = 0; i < line.length(); i++) {
            char ch = line.charAt(i);
            if (Character.isAlphabetic(ch)) set.add(ch);
        }
    }
    return new ArrayList<>(set);
}

static Set<Character> findLeadingLetters(List<String> lines) {
    Set<Character> leading = new HashSet<>();
    for (String line : lines) {
        List<String> tokens = tokensFromLine(line);
        List<String> relevant = tokens.stream().filter(t ->
!t.equals(".")).collect(Collectors.toList());
        if (relevant.size() <= 1) continue;
        for (String t : tokens) {
            if (t.equals(".")) continue;

```



<CODEWARS2026/>
LEÓN

```
        if (t.length() == 1 &&
Character.isAlphabetic(t.charAt(0))) {
            leading.add(t.charAt(0));
        }
        // si no es letra (por ejemplo, dígito literal), no
añadimos
        break;
    }
}
return leading;
}
// === Verificación completa para un mapeo ===
static boolean checkMapping(Map<Character, Character> mapping,
List<List<String>> linesTokens) {
    List<BigInteger> numbers = new ArrayList<>();
    List<String> numStrPerLine = new ArrayList<>(linesTokens.size());
    for (List<String> tokens : linesTokens) {
        String numstr = buildNumberFromTokens(tokens, mapping);
        numStrPerLine.add(numstr);
        if (numstr.isEmpty()) {
            // línea sin números (posible), ignoramos en 'numbers'
            continue;
        }
        if (numstr.length() > 1 && numstr.charAt(0) == '0') {
            // cero a la izquierda
            return false;
        }
    }
}
```



```
        if (!numstr.chars().allMatch(Character::isDigit)) return
false;

        numbers.add(new BigInteger(numstr));
    }
    if (numbers.size() < 3) return false;
    BigInteger multiplicand1 = numbers.get(0);
    BigInteger multiplicand2 = numbers.get(1);
    BigInteger finalResult = numbers.get(numbers.size() - 1);
    if (!multiplicand1.multiply(multiplicand2).equals(finalResult))
return false;

    // === Verificación de productos parciales con alineado por puntos
===

    int totalColumns = linesTokens.get(linesTokens.size() - 1).size();
    if (totalColumns <= 0) return false;
    List<List<String>> partialLines = new ArrayList<>();
    if (linesTokens.size() >= 4) { // 2 factores + >=1 parcial +
resultado
        for (int i = 2; i + 1 < linesTokens.size(); i++) {
            partialLines.add(linesTokens.get(i));
        }
    }
    String m2Str = buildNumberFromTokens(linesTokens.get(1), mapping);
    if (m2Str.isEmpty()) return false;
    if (partialLines.size() != m2Str.length()) return false;
    for (int i = 0; i < partialLines.size(); i++) {
        List<String> partialTokens = partialLines.get(i);
        int digit = Character.digit(m2Str.charAt(m2Str.length() - 1 -
i), 10);
```



<CODEWARS2026/>
LEÓN

```
        if (digit < 0) return false;
        String expectedValueStr =
multiplicand1.multiply(BigInteger.valueOf(digit)).toString();
        int expectedLeftDots = totalColumns -
(expectedValueStr.length() + i);
        int actualLeftDots = countLeadingDots(partialTokens);
        String actualNumStr = buildNumberFromTokens(partialTokens,
mapping);
        if (digit == 0) {
            boolean onlyDots = partialTokens.stream().allMatch(t ->
t.equals("."));
            if (onlyDots) {
                continue; // válido: parcial cero como "todo puntos"
            } else {
                if (!(actualNumStr.equals("0") && actualLeftDots ==
expectedLeftDots)) {
                    return false;
                }
            }
        } else {
            if (!(actualNumStr.equals(expectedValueStr) &&
actualLeftDots == expectedLeftDots)) {
                return false;
            }
        }
    }
    return true;
}
```

<let's code/>





<CODEWARS2026/>
LEÓN

```
// === Backtracking: asignar letras -> dígitos ===
static final class Solver {
    final List<String> lines;
    final List<List<String>> linesTokens;
    final List<Character> letters;        // ordenadas
    final Set<Character> leadingLetters;
    final Map<Character, Character> mapping = new HashMap<>();
    final boolean[] used = new boolean[10];
    Solver(List<String> lines) {
        this.lines = lines;
        this.linesTokens = new ArrayList<>();
        for (String ln : lines)
this.linesTokens.add(tokensFromLine(ln));
        this.letters = extractLetters(lines);
        this.leadingLetters = findLeadingLetters(lines);
    }
    boolean dfsAssign(int idx) {
        if (idx == letters.size()) {
            return checkMapping(mapping, linesTokens);
        }
        char L = letters.get(idx);
        for (int d = 0; d <= 9; d++) {
            if (used[d]) continue;
            if (d == 0 && leadingLetters.contains(L)) continue;
            mapping.put(L, (char) ('0' + d));
            used[d] = true;
            if (dfsAssign(idx + 1)) return true;
            used[d] = false;
        }
    }
}
```

```
        mapping.remove(L);
    }
    return false;
}

boolean run(Map<Character, Character> outMapping) {
    if (letters.size() > 10) return false;
    boolean ok = dfsAssign(0);
    if (ok) outMapping.putAll(mapping);
    return ok;
}

}

public static void main(String[] args) throws Exception {
    BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
    List<String> lines = parseInput(br);
    Solver solver = new Solver(lines);
    Map<Character, Character> mapping = new HashMap<>();
    if (solver.run(mapping)) {
        // imprimir en orden alfabético: A=6;B=3;...
        List<Character> keys = new ArrayList<>(solver.letters); // ya
ordenadas

        StringBuilder out = new StringBuilder();
        for (int i = 0; i < keys.size(); i++) {
            char k = keys.get(i);
            out.append(k).append('=').append(mapping.get(k));
            if (i + 1 < keys.size()) out.append(';');
        }
        System.out.println(out);
    }
}
```




<CODEWARS2026/>
LEÓN

```
    } else {  
        System.out.println("No solution found");  
    }  
}  
}
```

25

The eternal calendar

14 points

Introduction

The Forge of Chronos: The Creation of the Eternal Calendar

Long ago, before time had shape and the seasons were but a chaotic dance of the elements, the gods convened on Mount Aetherion, the sacred peak where the threads of existence were woven. The world was young, and mortals struggled to make sense of the cycles of light and shadow, of warmth and frost. It was then that Chronos, the primordial god of time, decreed that order must be brought to the chaos.

He summoned his three divine children to undertake this monumental task: Eirene, goddess of peace and balance; Helion, god of the blazing sun; and Seleneia, goddess of the silver moon. Together, they were entrusted with crafting a system that would guide mortals through the endless flow of time—a calendar that would forever mark the rhythm of life.

The Council of Seasons

The gods debated for seven days and nights beneath the great Celestial Oak, whose leaves shimmered with the stars. Each deity brought their perspective.

- Eirene spoke first: "Let the calendar be a tapestry of harmony. Each season should reflect balance between light and dark, growth and rest."
- Helion, his golden hair blazing like a summer sun, countered: "No! The calendar must honor the sun's power. Mortals should celebrate its rise and lament its fall."
- Seleneia, draped in a cloak of stars, whispered: "The moon, too, has its place. Its cycles have long guided tides and dreams. We must weave its phases into this creation."

But their visions clashed, and unity seemed impossible. To resolve this, Chronos ordered them to descend to the mortal world and seek inspiration.

The Trial of Mortals

Eirene journeyed to the lush meadows of Elara, where mortals tended crops and prayed for bountiful harvests. She witnessed how the changing seasons brought both prosperity and hardship, teaching her that balance was indeed the foundation of life.

Helion descended into the desert of Pyraxia, where mortals worshiped the sun as a giver of life and feared its absence. He realized the sun's journey across the sky could mark the passage of days and inspire celebration.

Seleneia roamed the shores of Aqualis, where fishermen guided their boats by the moonlight. She understood how the moon's phases mirrored the flow of time, its waxing and waning offering a cycle of hope and renewal.

The Forge of Chronos

Returning to Mount Aetherion, the siblings now worked together, their lessons from the mortal realm guiding their hands. They toiled in the Forge of Chronos, a mythical place where time itself flowed like molten silver.

- Eirene divided the year into four harmonious seasons, each reflecting the balance of growth and rest.
- Helion crafted the solar arc, a golden ring marking the days of the year as the sun journeyed through the sky.
- Seleneia inscribed the phases of the moon onto an obsidian disc, ensuring that its cycles would guide the months.

Chronos, with his immense power, bound these elements together into a single artifact: the *Eternal Calendar*. Its face shimmered with solar radiance, lunar grace, and seasonal wisdom, a perfect union of their divine wills. The Eternal calendar was created with the following logo:



The Calendar's Legacy

When the gods presented the Eternal Calendar to the mortals, they rejoiced. Farmers could now sow and reap with certainty, sailors could navigate by moonlight, and all could celebrate the changing seasons with festivals of light, harvest, and renewal.

From that day forward, the Eternal Calendar not only guided mortals but also became a bridge between gods and humanity, reminding all of the delicate balance that sustains existence.

Chronos, watching from his throne, smiled. Time had been given shape, and with it, the world could thrive in harmony. Thus, the Eternal Calendar became the greatest legacy of the gods, its creation an epic tale of discord, unity, and divine purpose.

Exercise

Write a program that receives as input a year and returns the calendar for given year.

Input

The program will simply receive a year. For instance:

2024

Output

And it will output that's years calendar. Something like:

2024																				
January							February							March						
Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa
	1	2	3	4	5	6					1	2	3						1	2
7	8	9	10	11	12	13	4	5	6	7	8	9	10	3	4	5	6	7	8	9
14	15	16	17	18	19	20	11	12	13	14	15	16	17	10	11	12	13	14	15	16
21	22	23	24	25	26	27	18	19	20	21	22	23	24	17	18	19	20	21	22	23
28	29	30	31				25	26	27	28	29			24	25	26	27	28	29	30
														31						
April							May							June						
Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa
	1	2	3	4	5	6					1	2	3							1
7	8	9	10	11	12	13	5	6	7	8	9	10	11	2	3	4	5	6	7	8
14	15	16	17	18	19	20	12	13	14	15	16	17	18	9	10	11	12	13	14	15
21	22	23	24	25	26	27	19	20	21	22	23	24	25	16	17	18	19	20	21	22
28	29	30					26	27	28	29	30	31		23	24	25	26	27	28	29
														30						
July							August							September						
Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa
	1	2	3	4	5	6					1	2	3	1	2	3	4	5	6	7
7	8	9	10	11	12	13	4	5	6	7	8	9	10	8	9	10	11	12	13	14
14	15	16	17	18	19	20	11	12	13	14	15	16	17	15	16	17	18	19	20	21
21	22	23	24	25	26	27	18	19	20	21	22	23	24	22	23	24	25	26	27	28
28	29	30	31				25	26	27	28	29	30	31	29	30					
October							November							December						
Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa
		1	2	3	4	5						1	2	1	2	3	4	5	6	7
6	7	8	9	10	11	12	3	4	5	6	7	8	9	8	9	10	11	12	13	14
13	14	15	16	17	18	19	10	11	12	13	14	15	16	15	16	17	18	19	20	21
20	21	22	23	24	25	26	17	18	19	20	21	22	23	22	23	24	25	26	27	28
27	28	29	30	31			24	25	26	27	28	29	30	29	30	31				

Example 1

Input

2024

Output

2024																				
January							February							March						
Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa
	1	2	3	4	5	6					1	2	3						1	2
7	8	9	10	11	12	13	4	5	6	7	8	9	10	3	4	5	6	7	8	9
14	15	16	17	18	19	20	11	12	13	14	15	16	17	10	11	12	13	14	15	16
21	22	23	24	25	26	27	18	19	20	21	22	23	24	17	18	19	20	21	22	23
28	29	30	31				25	26	27	28	29			24	25	26	27	28	29	30
														31						
April							May							June						
Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa
	1	2	3	4	5	6					1	2	3							1
7	8	9	10	11	12	13	5	6	7	8	9	10	11	2	3	4	5	6	7	8
14	15	16	17	18	19	20	12	13	14	15	16	17	18	9	10	11	12	13	14	15
21	22	23	24	25	26	27	19	20	21	22	23	24	25	16	17	18	19	20	21	22
28	29	30					26	27	28	29	30	31		23	24	25	26	27	28	29
														30						
July							August							September						
Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa
	1	2	3	4	5	6					1	2	3	1	2	3	4	5	6	7
7	8	9	10	11	12	13	4	5	6	7	8	9	10	8	9	10	11	12	13	14
14	15	16	17	18	19	20	11	12	13	14	15	16	17	15	16	17	18	19	20	21
21	22	23	24	25	26	27	18	19	20	21	22	23	24	22	23	24	25	26	27	28
28	29	30	31				25	26	27	28	29	30	31	29	30					
October							November							December						
Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa
		1	2	3	4	5						1	2	1	2	3	4	5	6	7
6	7	8	9	10	11	12	3	4	5	6	7	8	9	8	9	10	11	12	13	14
13	14	15	16	17	18	19	10	11	12	13	14	15	16	15	16	17	18	19	20	21
20	21	22	23	24	25	26	17	18	19	20	21	22	23	22	23	24	25	26	27	28
27	28	29	30	31			24	25	26	27	28	29	30	29	30	31				

Solutions

Python

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import sys
# Parámetros globales (equivalentes a las globals del C)
```



<CODEWARS2026/> LEÓN

```
width = 80
year = None
cols = 0
lead = 0
gap = 0
wdays = ["Su", "Mo", "Tu", "We", "Th", "Fr", "Sa"]
# months: name, days, start_wday, at
months = [
    {"name": "January",    "days": 31, "start_wday": 0, "at": 0},
    {"name": "February",  "days": 28, "start_wday": 0, "at": 0},
    {"name": "March",      "days": 31, "start_wday": 0, "at": 0},
    {"name": "April",      "days": 30, "start_wday": 0, "at": 0},
    {"name": "May",        "days": 31, "start_wday": 0, "at": 0},
    {"name": "June",       "days": 30, "start_wday": 0, "at": 0},
    {"name": "July",       "days": 31, "start_wday": 0, "at": 0},
    {"name": "August",     "days": 31, "start_wday": 0, "at": 0},
    {"name": "September",  "days": 30, "start_wday": 0, "at": 0},
    {"name": "October",    "days": 31, "start_wday": 0, "at": 0},
    {"name": "November",   "days": 30, "start_wday": 0, "at": 0},
    {"name": "December",   "days": 31, "start_wday": 0, "at": 0},
]

def space(n: int) -> None:
    """Imprime n espacios."""
    if n > 0:
        sys.stdout.write(" " * n)

def init_months() -> None:
    """Inicializa días de febrero si bisiesto, inicio de semana por mes y
    layout."""
```



<CODEWARS2026/> LEÓN

```
global year, cols, gap, lead, months

# Ajuste por año bisiesto: (divisible por 4 y no por 100) o divisible
por 400
if ((year % 4 == 0 and year % 100 != 0) or (year % 400 == 0)):
    months[1]["days"] = 29
else:
    months[1]["days"] = 28

# Cálculo del día de la semana del 1 de enero (Sunday=0)
y = year - 1
months[0]["start_wday"] = (y * 365 + y // 4 - y // 100 + y // 400 + 1)
% 7

# Inicio de semana del resto de meses
for i in range(1, 12):
    months[i]["start_wday"] = (months[i - 1]["start_wday"] + months[i
- 1]["days"]) % 7

# Configurar columnas según ancho
cols = (width + 2) // 22
while 12 % cols != 0:
    cols -= 1

gap = ((width - 20 * cols) // (cols - 1)) if (cols - 1) > 0 else 0
if gap > 4:
    gap = 4

lead = (width - (20 + gap) * cols + gap + 1) // 2

def print_row(row: int) -> None:
    """Imprime una fila de 'cols' meses con encabezado y semanas."""
    global cols, lead, gap, months
    from_idx = row * cols
    to_idx = from_idx + cols
```




<CODEWARS2026/>
LEÓN

```
# Títulos de los meses (centrados en 20)
space(lead)
for c in range(from_idx, to_idx):
    name = months[c]["name"]
    i = len(name)
    left = (20 - i) // 2
    space(left)
    sys.stdout.write(name)
    # Relleno derecho + gap entre meses (no añadir gap al último)
    right = 20 - i - left + (0 if c == to_idx - 1 else gap)
    space(right)
sys.stdout.write("\n")
# Encabezado de días
space(lead)
for c in range(from_idx, to_idx):
    for i in range(7):
        sys.stdout.write(wdays[i])
        if i != 6:
            sys.stdout.write(" ")
    if c < to_idx - 1:
        space(gap)
    else:
        sys.stdout.write("\n")
# Semanas (días)
while True:
    # Comprobar si quedan días por imprimir en esta fila
    any_remaining = False
    for c in range(from_idx, to_idx):
```



<CODEWARS2026/>
LEÓN

```
        if months[c]["at"] < months[c]["days"]:
            any_remaining = True
            break
    if not any_remaining:
        break
    space(lead)
    for c in range(from_idx, to_idx):
        i = 0
        # Desplazamiento inicial según start_wday
        for _ in range(months[c]["start_wday"]):
            space(3)
            i += 1
        # Imprimir hasta completar la semana o agotar días del mes
        while i < 7 and months[c]["at"] < months[c]["days"]:
            months[c]["at"] += 1
            sys.stdout.write(f"{months[c]['at']:2d}")
            i += 1
            # Espacio entre días o entre meses
            if i < 7 or c < to_idx - 1:
                sys.stdout.write(" ")
        # Relleno hasta completar los 7 días visibles de la línea para
este mes
        while i < 7 and c < to_idx - 1:
            space(3)
            i += 1
        # Espaciado extra entre meses (ajustado)
        if c < to_idx - 1:
            space(gap - 1)
```



<CODEWARS2026/> LEÓN

```
# Tras la primera línea de semanas, el desplazamiento se anula
months[c]["start_wday"] = 0
sys.stdout.write("\n")
# Línea en blanco entre filas (excepto la última con diciembre)
if from_idx + cols < 12:
    sys.stdout.write("\n")
def print_year() -> None:
    """Imprime el año centrado y todas las filas de meses."""
    global year, width, cols
    buf = str(year)
    space((width - len(buf)) // 2)
    sys.stdout.write(buf + "\n\n")
    row = 0
    while row * cols < 12:
        print_row(row)
        row += 1
def bail(argv0: str) -> None:
    """Imprime uso y sale con error."""
    sys.stderr.write(f"bad args\nUsage: {argv0} year [-w width (>=
20)]\n")
    sys.exit(1)
def main() -> None:
    global width, year, months
    argv = sys.argv
    argc = len(argv)
    # Si hay año por argumento, úsalo; si no, intenta leer de stdin.
    year_set = False
    # Parseo de argumentos al estilo del C
```



<CODEWARS2026/> LEÓN

```
i = 1
while i < argc:
    if argv[i] == "-w":
        i += 1
        if i == argc:
            bail(argv[0])
        try:
            width = int(argv[i])
        except ValueError:
            bail(argv[0])
        if width < 20:
            bail(argv[0])
    else:
        # Primer argumento no -w lo tratamos como año
        if not year_set:
            try:
                y = int(argv[i])
            except ValueError:
                y = 1969
            if y <= 0:
                y = 1969
            year = y
            year_set = True
        else:
            bail(argv[0])
    i += 1
# Si no se estableció año por argv, intenta leer de stdin
if not year_set:
```




<CODEWARS2026/> LEÓN

```
try:
    line = sys.stdin.readline()
    if line:
        y = int(line.strip())
        if y <= 0:
            y = 1969
        year = y
    else:
        # Sin entrada: usar valor por defecto del C
        year = 1969
except Exception:
    year = 1969

# Seguridad: reiniciar 'at' por si se reutiliza el módulo
for m in months:
    m["at"] = 0
init_months()
print_year()

if __name__ == "__main__":
    main()
```

C++

```
//#Source: shorturl.at/adg05
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int width = 80, year;
int cols, lead, gap;
const char *wdays[] = { "Su", "Mo", "Tu", "We", "Th", "Fr", "Sa" };
struct months {
```



<CODEWARS2026/> LEÓN

```
const char *name;
int days, start_wday, at;
} months[12] = {
    { "January",    31, 0, 0 },
    { "February",   28, 0, 0 },
    { "March",      31, 0, 0 },
    { "April",      30, 0, 0 },
    { "May",        31, 0, 0 },
    { "June",       30, 0, 0 },
    { "July",       31, 0, 0 },
    { "August",     31, 0, 0 },
    { "September",  30, 0, 0 },
    { "October",    31, 0, 0 },
    { "November",   30, 0, 0 },
    { "December",   31, 0, 0 }
};

void space(int n) { while (n-- > 0) putchar(' '); }
void init_months()
{
    int i;
    if (!(year % 4) && (year % 100) || !(year % 400))
        months[1].days = 29;
    year--;
    months[0].start_wday
        = (year * 365 + year/4 - year/100 + year/400 + 1) % 7;
    for (i = 1; i < 12; i++)
        months[i].start_wday =
            (months[i-1].start_wday + months[i-1].days) % 7;
```



<CODEWARS2026/>
LEÓN

```
cols = (width + 2) / 22;
while (12 % cols) cols--;
gap = cols - 1 ? (width - 20 * cols) / (cols - 1) : 0;
if (gap > 4) gap = 4;
lead = (width - (20 + gap) * cols + gap + 1) / 2;
    year++;
}
void print_row(int row)
{
    int c, i, from = row * cols, to = from + cols;
    space(lead);
    for (c = from; c < to; c++) {
        i = strlen(months[c].name);
        space((20 - i)/2);
        printf("%s", months[c].name);
        space(20 - i - (20 - i)/2 + ((c == to - 1) ? 0 : gap));
    }
    putchar('\n');
    space(lead);
    for (c = from; c < to; c++) {
        for (i = 0; i < 7; i++)
            printf("%s%s", wdays[i], i == 6 ? "" : " ");
        if (c < to - 1) space(gap);
        else putchar('\n');
    }
    while (1) {
        for (c = from; c < to; c++)
            if (months[c].at < months[c].days) break;
    }
}
```

<let's code/>





<CODEWARS2026/>
LEÓN

```
        if (c == to) break;
        space(lead);
        for (c = from; c < to; c++) {
            for (i = 0; i < months[c].start_wday; i++) space(3);
            while(i++ < 7 && months[c].at < months[c].days) {
                printf("%2d", ++months[c].at);
                if (i < 7 || c < to - 1) putchar(' ');
            }
            while (i++ <= 7 && c < to - 1) space(3);
            if (c < to - 1) space(gap - 1);
            months[c].start_wday = 0;
        }
        putchar('\n');
    }
    // Check if it's the last row (with December) and don't add an extra
    line break.
    if (from + cols < 12) {
        putchar('\n');
    }
}

void print_year()
{
    int row;
    char buf[32];
    sprintf(buf, "%d", year);
    space((width - strlen(buf)) / 2);
    printf("%s\n\n", buf);
    for (row = 0; row * cols < 12; row++)
```




<CODEWARS2026/>
LEÓN

```
        print_row(row);
    }
    int main(int c, char **v)
    {
        int i, year_set = 0;
        scanf("%d", &year);
        for (i = 1; i < c; i++) {
            if (!strcmp(v[i], "-w")) {
                if (++i == c || (width = atoi(v[i])) < 20)
                    goto bail;
            } else if (!year_set) {
                if (!sscanf(v[i], "%d", &year) || year <= 0)
                    year = 1969;
                year_set = 1;
            } else
                goto bail;
        }
        init_months();
        print_year();
        return 0;
    bail: fprintf(stderr, "bad args\nUsage: %s year [-w width (>= 20)]\n",
v[0]);
        exit(1);
    }
```

Java

```
import java.util.Scanner;
public class Main {
    // Parámetros globales (equivalentes a las globals del C)
```



<CODEWARS2026/> LEÓN

```
static int width = 80, year;
static int cols, lead, gap;
static final String[] wdays = { "Su", "Mo", "Tu", "We", "Th", "Fr",
"Sa" };
static class Month {
    final String name;
    int days;
    int startWday;
    int at; // contador de días impresos
    Month(String name, int days) {
        this.name = name;
        this.days = days;
        this.startWday = 0;
        this.at = 0;
    }
}

static final Month[] months = {
    new Month("January", 31),
    new Month("February", 28),
    new Month("March", 31),
    new Month("April", 30),
    new Month("May", 31),
    new Month("June", 30),
    new Month("July", 31),
    new Month("August", 31),
    new Month("September", 30),
    new Month("October", 31),
    new Month("November", 30),
```



<CODEWARS2026/>
LEÓN

```
        new Month("December", 31)
    };
    static void space(int n) {
        if (n <= 0) return;
        for (int i = 0; i < n; i++) System.out.print(' ');
    }
    static void initMonths() {
        // Año bisiesto: (divisible por 4 y no por 100) o divisible por
400
        if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0))
            months[1].days = 29;
        else
            months[1].days = 28;
        // Día de la semana del 1 de enero (Sunday=0) usando el mismo
cómputo que el C
        int y = year - 1;
        months[0].startWday = (y * 365 + y / 4 - y / 100 + y / 400 + 1) %
7;

        // Inicio de semana del resto de meses (cascada)
        for (int i = 1; i < 12; i++) {
            months[i].startWday = (months[i - 1].startWday + months[i -
1].days) % 7;
        }
        // Configurar columnas según ancho (replica el C)
        cols = (width + 2) / 22;
        while (12 % cols != 0) cols--;
        gap = (cols - 1) > 0 ? (width - 20 * cols) / (cols - 1) : 0;
        if (gap > 4) gap = 4;
```



<CODEWARS2026/>
LEÓN

```
        lead = (width - (20 + gap) * cols + gap + 1) / 2;
        // Reiniciar contadores "at" por seguridad
        for (Month m : months) m.at = 0;
    }
    static void printRow(int row) {
        int from = row * cols;
        int to = from + cols;
        // Títulos de los meses (centrados en 20)
        space(lead);
        for (int c = from; c < to; c++) {
            String name = months[c].name;
            int i = name.length();
            int left = (20 - i) / 2;
            space(left);
            System.out.print(name);
            int right = 20 - i - left + ((c == to - 1) ? 0 : gap);
            space(right);
        }
        System.out.println();
        // Encabezado de días
        space(lead);
        for (int c = from; c < to; c++) {
            for (int i = 0; i < 7; i++) {
                System.out.print(wdays[i]);
                if (i != 6) System.out.print(' ');
            }
            if (c < to - 1) space(gap);
            else System.out.println();
        }
    }
}
```




<CODEWARS2026/>
LEÓN

```
    }  
    // Semanas (días)  
    while (true) {  
        boolean anyRemaining = false;  
        for (int c = from; c < to; c++) {  
            if (months[c].at < months[c].days) { anyRemaining = true;  
break; }  
        }  
        if (!anyRemaining) break;  
        space(lead);  
        for (int c = from; c < to; c++) {  
            int i = 0;  
            // Desplazamiento inicial según startWday  
            for (int s = 0; s < months[c].startWday; s++) {  
                space(3);  
                i++;  
            }  
            // Imprimir hasta completar la semana o agotar días del  
mes  
            while (i < 7 && months[c].at < months[c].days) {  
                months[c].at++;  
                System.out.printf("%2d", months[c].at);  
                i++;  
                // Espacio entre días o entre meses  
                if (i < 7 || c < to - 1) System.out.print(' ');  
            }  
            // Relleno hasta completar los 7 "slots" visibles de la  
línea para este mes
```



<CODEWARS2026/>
LEÓN

```
        while (i < 7 && c < to - 1) {
            space(3);
            i++;
        }
        // Espaciado extra entre meses (ajustado)
        if (c < to - 1) space(gap - 1);
        // Tras la primera línea del mes, anular desplazamiento
        months[c].startWday = 0;
    }
    System.out.println();
}
// Línea en blanco entre filas (excepto la última con diciembre)
if (from + cols < 12) System.out.println();
}
static void printYear() {
    String buf = Integer.toString(year);
    space((width - buf.length()) / 2);
    System.out.println(buf);
    System.out.println();
    for (int row = 0; row * cols < 12; row++) {
        printRow(row);
    }
}
static void bail(String prog) {
    System.err.printf("bad args%nUsage: %s year [-w width (>= 20)]%n",
prog);
    System.exit(1);
}
```



<CODEWARS2026/>
LEÓN

```
public static void main(String[] args) {  
    String prog = Main.class.getSimpleName();  
    // Primero: intentar leer 'year' de stdin como en el C  
    Integer stdinYear = null;  
    Scanner sc = new Scanner(System.in);  
    if (sc.hasNextInt()) {  
        stdinYear = sc.nextInt();  
    }  
    boolean yearSet = false;  
    // Parseo de argumentos al estilo del C  
    for (int i = 0; i < args.length; i++) {  
        if ("-w".equals(args[i])) {  
            i++;  
            if (i == args.length) bail(prog);  
            try {  
                width = Integer.parseInt(args[i]);  
            } catch (NumberFormatException e) {  
                bail(prog);  
            }  
            if (width < 20) bail(prog);  
        } else if (!yearSet) {  
            int y;  
            try {  
                y = Integer.parseInt(args[i]);  
            } catch (NumberFormatException e) {  
                y = 1969;  
            }  
            if (y <= 0) y = 1969;  
        }  
    }  
}
```

<let's code/>





<CODEWARS2026/>
LEÓN

```
        year = y;
        yearSet = true;
    } else {
        bail(prog);
    }
}

// Si no se estableció año por argv, usar el leído de stdin; si no
hay, usar 1969
if (!yearSet) {
    if (stdinYear != null && stdinYear > 0) {
        year = stdinYear;
    } else {
        year = 1969;
    }
}
initMonths();
printYear();
}
}
```

<let's code/>



1

Welcome to Codewars I

1 points

Introduction

Now is the time to prove that you have what it takes to be a true warrior. Now is the time to start coding and having fun! As a rite of passage, you must create your very first program in this journey. However, we are feeling generous today, so we will not ask you to do something too difficult. For now you only have to write a program that takes no input and returns the string "Welcome to Codewars 2026!".

Exercise

Write a program that outputs the string "Welcome to Codewars 2026!".

Input

None.

Output

The string "Welcome to Codewars 2026!".

Example 1

Input

No input for this problem.

Output

```
Welcome to Codewars 2026!
```



<CODEWARS2026/>
LEÓN

Solutions

Python

```
def main():  
    print("Welcome to Codewars 2026!")  
if __name__ == "__main__":  
    main()
```

C++

```
#include <iostream>  
int main(void) {  
    std::cout << "Welcome to Codewars 2026!" << std::endl;  
    return 0;  
}
```

Java

```
class Welcome {  
    public static void main(String[] args) {  
        System.out.println("Welcome to Codewars 2026!");  
    }  
}
```

2

Welcome to Codewars II

2 points

Introduction

Handling input will also be important for the challenges you will face today. Now, you have to extend your first program a bit, so it takes a team name (like the name of your team, but it can be any arbitrary string) and outputs the string *Welcome to Codewars 2026, TeamName!*, where *TeamName* is the name of the team you input.

Exercise

Write a program that outputs the string *Welcome to Codewars 2026, TeamName!* where *TeamName* is the name of the team you input.

Input

A string representing a name of a team.

Output

The string *Welcome to Codewars 2026, TeamName!* where *TeamName* is the name of the team you input.

Example 1

Input

```
Trambolikos
```

Output

```
Welcome to Codewars 2026, Trambolikos!
```

Example 2

Input

Los top del Mundo

Output

Welcome to Codewars 2026, Los top del Mundo!

Solutions

Python

```
def main():  
    team = input()  
    print("Welcome to Codewars 2026, " + team + "!")  
if __name__ == "__main__":  
    main()
```

C++

```
#include <iostream>  
#include <string>  
int main(void) {  
    std::string team;  
    std::getline(std::cin, team);  
    std::cout << "Welcome to Codewars 2026, " << team << "!" << std::endl;  
    return 0;  
}
```

Java

```
import java.util.*;
```




<CODEWARS2026/>
LEÓN

```
class Welcome2 {  
    public static void main(String[] args) {  
        Scanner scanner = new Scanner(System.in);  
        String teamName = scanner.nextLine().trim();  
        System.out.println("Welcome to Codewars 2026, " + teamName+"!");  
    }  
}
```

<let's code/>



3

Reverse DNA

4 points

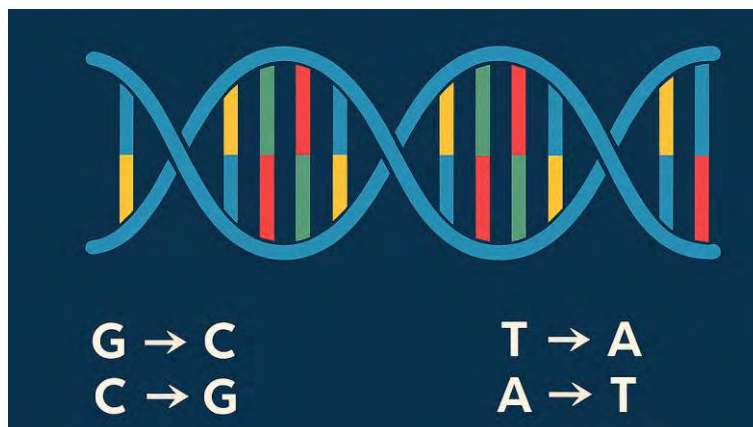
Introduction

You have been recently hired by a pharmaceutical company and are understanding the basics on heredability. One of your first tasks in the company is to understand that DNA (deoxyribonucleic acid) is the molecule that encapsulates all the information of the life species.

The encoding is done through four nucleotides: Adenine, Thymine, Cytosine and Guanine that are usually abbreviated as A, T, C, G. In each position of one DNA strand, we can have just one of these nucleotides and the paired strand codifies the opposite base (A links with T, C with G) so the molecule encodes two bits per base pair.

Therefore for each base, their corresponding pair is:

G → C
C → G
T → A
A → T



The complement of a DNA sequence is the sequence you get by replacing each base by its pair.

An equivalent protein called "reverse complement", is obtained if you find the complement of a sequence and then reverse it.

Exercise

You are asked to write a program that finds the reverse complement of a DNA sequence.

Input

The input of the program will be a list of sequences of nucleotide bases, each in one line, ending with a line with a single dot ('.'). As it's your first project, nucleotide sequences will always be shorter than 100 nucleotides (expressed as characters). An example of such input would be:

```
AAAACCCGGT
GCGCTTA
.
```

Output

Given the corresponding pairs of each nucleotide, your program will have to write the "reverse complement" of each nucleotide sequence in the list (in the same order). For the previous input, the expected output would be:

```
ACCGGGTTTT
TAAGCGC
```

Example 1

Input

```
A
```



<CODEWARS2026/> LEÓN

```
AC
ACG
ACGT
ACGAAGCGATAGCTGACTAGCTAGCATGCTAATCTAGA
.
```

Output

```
T
GT
CGT
ACGT
TCTAGATTAGCATGCTAGCTAGTCAGCTATCGCTTCGT
```

Example 2

Input

```
AAACACG
ACGTACGAAGCGATAGCTGACTAGCTAGCATGCTAATCTAGAACACG
.
```

Output

```
CGTGTTT
CGTGTTCTAGATTAGCATGCTAGCTAGTCAGCTATCGCTTCGTACGT
```

Solutions

Python

```
#!/usr/bin/env python3
import sys
# Tabla de complementos DNA
COMP = str.maketrans({
    'A': 'T',
    'T': 'A',
    'C': 'G',
```




<CODEWARS2026/> LEÓN

```
'G': 'C',
})
def reverse_complement(seq: str) -> str:
    # Complementa y revierte
    return seq.translate(COMP[::-1])
def main():
    out_lines = []
    for line in sys.stdin:
        s = line.strip()
        if not s:
            # Si hubiera líneas vacías, las ignoramos (no se especifican,
            # pero no molesta)
            continue
        if s == '.':
            break
        out_lines.append(reverse_complement(s))
    sys.stdout.write("\n".join(out_lines))
    if out_lines:
        sys.stdout.write("\n")
if __name__ == "__main__":
    main()
```

C++

```
#include <iostream>
#include <string>
#include <array>
#include <stdexcept>
static inline char complement(char b) {
    switch (b) {
```



<CODEWARS2026/>
LEÓN

```
        case 'A': return 'T';
        case 'T': return 'A';
        case 'C': return 'G';
        case 'G': return 'C';
        default:
            throw std::runtime_error(std::string("Base invalida: ") + b);
    }
}

int main() {
    std::ios::sync_with_stdio(false);
    std::cin.tie(nullptr);
    std::string s;
    while (std::getline(std::cin, s)) {
        // Por si el input viene con CRLF (Windows)
        if (!s.empty() && s.back() == '\\r') s.pop_back();
        if (s == ".") break;
        if (s.empty()) continue;
        // reverse + complement en una pasada
        for (int i = static_cast<int>(s.size()) - 1; i >= 0; --i) {
            std::cout << complement(s[i]);
        }
        std::cout << '\\n';
    }
    return 0;
}
```

Java

```
import java.io.BufferedReader;
import java.io.IOException;
```



<CODEWARS2026/>
LEÓN

```
import java.io.InputStreamReader;

public class dna {
    private static char complement(char base) {
        switch (base) {
            case 'A': return 'T';
            case 'T': return 'A';
            case 'C': return 'G';
            case 'G': return 'C';
            default:
                // Según el enunciado solo habrá A,C,G,T.
                // Si quieres hacerlo más robusto, puedes lanzar excepción
                // o devolver el mismo char.
                throw new IllegalArgumentException("Base inválida: " +
base);
        }
    }

    private static String reverseComplement(String seq) {
        StringBuilder sb = new StringBuilder(seq.length());
        // Recorremos desde el final: reverse + complement en una pasada
        for (int i = seq.length() - 1; i >= 0; i--) {
            sb.append(complement(seq.charAt(i)));
        }
        return sb.toString();
    }

    public static void main(String[] args) throws Exception {
        BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
        String line;
```



<CODEWARS2026/>
LEÓN

```
StringBuilder output = new StringBuilder();
while ((line = br.readLine()) != null) {
    line = line.trim();
    if (line.equals(".")) break;
    if (line.isEmpty()) continue; // por si acaso
    output.append(reverseComplement(line)).append('\n');
}
System.out.print(output.toString());
}
```

<let's code/>



4

The Galactic Alphabet Decoder

4 points



Introduction

Humanity has intercepted encrypted alien messages. Each letter in the message has been shifted forward by a constant value K in a cyclic alphabet ($A \rightarrow B \rightarrow C \dots \rightarrow Z \rightarrow A$). This means that after Z , the alphabet wraps around back to A .

Your task is to decode the message by shifting every letter backwards by K positions, restoring the original text. The backward shift is also cyclic: for example, shifting A backwards by 1 results in Z .

Exercise

Write a program that decodes alien encrypted messages.

Input

Your program will read two lines:

- An integer K ($0 \leq K \leq 25$)
- An uppercase string S (A-Z only)

Output

A single line containing the decoded message, obtained by shifting each letter of S backwards by K positions.

Example 1

Input

```
1  
BDF
```

Output

```
ACE
```

Example 2

Input

```
3  
KH00R
```

Output

```
HELLO
```

Solutions

Python



<CODEWARS2026/> LEÓN

```
def main():
    K = int(input())
    S = input().strip()
    result = []
    for c in S:
        pos = ord(c) - ord('A')      # Convertimos letra → número 0-25
        new_pos = (pos - K) % 26      # Retrocedemos K posiciones con ciclo
        result.append(chr(new_pos + ord('A')))
    print("".join(result))
if __name__ == "__main__":
    main()
```

C++

```
#include <iostream>
#include <string>
int main() {
    int K;
    std::string S;
    std::cin >> K;          // Leemos el desplazamiento
    std::cin >> S;          // Leemos la cadena (sin espacios)
    std::string result;
    result.reserve(S.size());
    for (char c : S) {
        int pos = c - 'A';      // Convertimos letra → número
        // 0-25
        int new_pos = (pos - K) % 26;    // Retrocedemos K posiciones
        // con ciclo
        // Aseguramos que el módulo sea positivo
        if (new_pos < 0) {
```




<CODEWARS2026/> LEÓN

```
        new_pos += 26;
    }
    result.push_back(static_cast<char>(new_pos + 'A'));
}
std::cout << result << std::endl;
return 0;
}
```

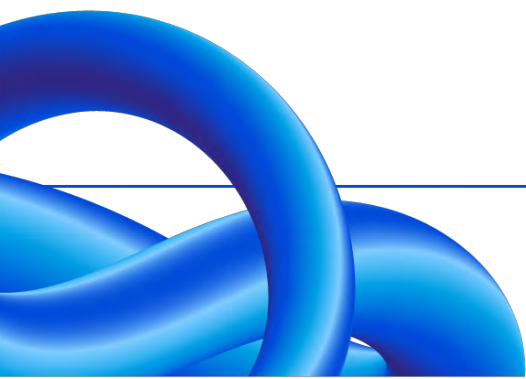
Java

```
import java.util.Scanner;
public class galactic {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int K = scanner.nextInt();    // Leemos el desplazamiento
        String S = scanner.next();    // Leemos la cadena (sin espacios)
        StringBuilder result = new StringBuilder();
        for (char c : S.toCharArray()) {
            int pos = c - 'A';          // Convertimos letra →
número 0-25
            int newPos = (pos - K) % 26;    // Retrocedemos K
posiciones con ciclo
            // Aseguramos que el módulo sea positivo
            if (newPos < 0) {
                newPos += 26;
            }
            result.append((char)(newPos + 'A'));
        }
        System.out.println(result.toString());
        scanner.close();
    }
}
```




<CODEWARS2026/>
LEÓN

}
}



<let's code/>



5

Mr. Burns infamous salary calculator

5 points

Introduction

Mr. Burns, the infamous owner of the Springfield Nuclear Power Plant, has decided to calculate the salaries of his employees using a custom formula. However, Mr. Burns wants to ensure that the calculations are performed accurately, following the proper operator precedence.



The formula he uses to calculate the salary is as follows:

$$\text{salary} = \text{base} + \text{hours} \times \text{rate} - \text{deductions} / \text{tax} + (\text{bonus} - \text{penalty}) \times \text{multiplier}$$

Where:

- base: The base salary of the employee.
- hours: The total number of hours worked by the employee.
- rate: The hourly wage.

- deductions: The total amount of deductions for the employee.
- tax: The tax rate applied to deductions.
- bonus: Any additional bonus the employee earns.
- penalty: Penalty amount for any misconduct or inefficiency.
- multiplier: A performance multiplier based on the employee's evaluation.

Exercise

Your task:

1. Write a program that allows Mr. Burns to input the values for base, hours, rate, deductions, tax, bonus, penalty, and multiplier.
2. Use the formula provided to calculate the employee's salary.
3. Print the final salary for Mr. Burns to review.

Remember! Each number must be used only once and all numbers must be greater than 0.

Take note that divisions will be always obtained with only 2 digit numbers.

Input

The input will consist in 8 numbers, each one entered on one line, that corresponds to each of the values used to calculate the salary:

```
2000
40
25
300
15
500
```



<CODEWARS2026/> LEÓN

100

1.2

Output

The output will show the calculated salary, with 2 digits:

3460.00

or

Please enter valid numeric values.

or

Division by zero occurred. Check the tax rate.

Example 1

Input

2000

40

25

300

15

500

100

1.2

Output

3460.00

Example 2

Input

2109

111

11

300



<CODEWARS2026/>
LEÓN

10
240
80
1.3

Output

3508.00

Solutions

Python

```
def calculate_salary(base, hours, rate, deductions, tax, bonus, penalty, multiplier):  
    """Calculates the employee's salary based on the provided formula."""  
    try:  
        # Apply the formula  
        salary = base + (hours * rate) - (deductions / tax) + ((bonus -  
penalty) * multiplier)  
        return salary  
    except ZeroDivisionError:  
        return "Error: Division by zero occurred. Check the tax rate."  
  
def main():  
    # Get user inputs  
    try:  
        base = float(input())  
        hours = float(input())  
        rate = float(input())  
        deductions = float(input())  
        tax = float(input())  
        bonus = float(input())  
        penalty = float(input())
```



<CODEWARS2026/>
LEÓN

```
multiplier = float(input())
if base<0 or hours<0 or rate <0 or deductions<0 or tax <0 or
bonus<0 or penalty<0 or multiplier<0:
    raise ValueError("")
# Calculate salary
salary = calculate_salary(base, hours, rate, deductions, tax,
bonus, penalty, multiplier)
# Display result
print(f"{salary:.2f}" if isinstance(salary, float) else salary)
except ValueError:
    print("Please enter valid numeric values.")
if __name__ == "__main__":
    main()
```

C++

```
#include <iostream>
#include <iomanip>
#include <stdexcept>
double calculate_salary(double base, double hours, double rate, double
deductions, double tax, double bonus, double penalty, double multiplier) {
    // Aplica la fórmula
    if (tax == 0) throw std::runtime_error("Error: Division by zero
occurred. Check the tax rate.");
    return base + (hours * rate) - (deductions / tax) + ((bonus - penalty)
* multiplier);
}
int main() {
    double base, hours, rate, deductions, tax, bonus, penalty, multiplier;
    try {
```



```
std::cin >> base >> hours >> rate >> deductions >> tax >> bonus >>
penalty >> multiplier;
if (base < 0 || hours < 0 || rate < 0 || deductions < 0 || tax < 0
|| bonus < 0 || penalty < 0 || multiplier < 0) {
    throw std::invalid_argument("");
}
double salary = calculate_salary(base, hours, rate, deductions,
tax, bonus, penalty, multiplier);
std::cout << std::fixed << std::setprecision(2) << salary <<
std::endl;
} catch (const std::invalid_argument&) {
    std::cout << "Please enter valid numeric values." << std::endl;
} catch (const std::runtime_error& e) {
    std::cout << e.what() << std::endl;
} catch (...) {
    std::cout << "Please enter valid numeric values." << std::endl;
}
return 0;
}
```

Java

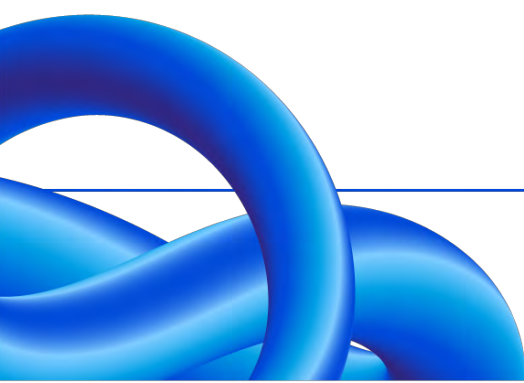
```
import java.util.Scanner;
public class Mr_Burns_infamous_salary_calculator {
    public static double calculateSalary(double base, double hours, double
rate, double deductions, double tax, double bonus, double penalty, double
multiplier) throws ArithmeticException {
        if (tax == 0) throw new ArithmeticException("Error: Division by
zero occurred. Check the tax rate.");
    }
}
```

```
        return base + (hours * rate) - (deductions / tax) + ((bonus -  
penalty) * multiplier);  
    }  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        try {  
            double base = Double.parseDouble(sc.nextLine());  
            double hours = Double.parseDouble(sc.nextLine());  
            double rate = Double.parseDouble(sc.nextLine());  
            double deductions = Double.parseDouble(sc.nextLine());  
            double tax = Double.parseDouble(sc.nextLine());  
            double bonus = Double.parseDouble(sc.nextLine());  
            double penalty = Double.parseDouble(sc.nextLine());  
            double multiplier = Double.parseDouble(sc.nextLine());  
            if (base < 0 || hours < 0 || rate < 0 || deductions < 0 || tax  
< 0 || bonus < 0 || penalty < 0 || multiplier < 0) {  
                throw new IllegalArgumentException();  
            }  
            double salary = calculateSalary(base, hours, rate, deductions,  
tax, bonus, penalty, multiplier);  
            System.out.printf("%.2f\n", salary);  
        } catch (IllegalArgumentException e) {  
            System.out.println("Please enter valid numeric values.");  
        } catch (ArithmeticException e) {  
            System.out.println("Error: Division by zero occurred. Check  
the tax rate.");  
        }  
    }  
}
```




<CODEWARS2026/>
LEÓN

}



<let's code/>



6

Pokemon Battle Effectiveness Simulator

5 points



Introduction

Ash Ketchum is preparing for the Pokemon World Tournament, and he needs your help! To win against tough opponents, Ash needs to understand how effective his Pokemon's moves are against different types.

Your task is to create a Pokemon Battle Effectiveness Simulator. Use the following type-effectiveness table to determine how strong an attack of one type is against a Pokemon of another type.

The numbers represent the attack's effectiveness:

#	Attacker\ Defender	Fire	Water	Grass	Electric	Ground	Rock	Flying	Ice	Psychic	Dragon
1	Fire	1.0	0.5	2.0	1.0	1.0	0.5	1.0	2.0	1.0	0.5

2	Water	2.0	1.0	0.5	1.0	2.0	2.0	1.0	1.0	1.0	0.5
3	Grass	0.5	2.0	1.0	1.0	2.0	1.0	0.5	1.0	1.0	0.5
4	Electric	1.0	1.0	2.0	1.0	0.0	1.0	2.0	1.0	1.0	0.5
5	Groun d	2.0	1.0	0.5	2.0	1.0	2.0	0.0	1.0	1.0	1.0
6	Rock	2.0	1.0	2.0	1.0	0.5	1.0	1.0	1.0	1.0	1.0
7	Flying	1.0	1.0	2.0	0.5	1.0	0.5	1.0	2.0	1.0	1.0
8	Ice	0.5	0.5	2.0	1.0	2.0	1.0	1.0	0.5	1.0	2.0
9	Psychic	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
10	Dragon	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	2.0

And the following is the table of texts that our program will show:

Value	Text to be shown
2.0	SUPER EFFECTIVE
1.0	NEUTRAL
0.5	NOT VERY EFFECTIVE
0.0	HAS NO EFFECT

Exercise

Write a program that:

- Receive the user selection of an attacker's type and a defender's type.
- Uses the table to determine the attack's effectiveness.
- Prints whether the attack is SUPER EFFECTIVE, NEUTRAL, NOT VERY EFFECTIVE, or HAS NO EFFECT.

Input

The input will consist in two lines with two positive integer numbers that represent the attacker's and defender's pokemon types. Therefore if we receive two numbers 1 and 3, the Attacker will be "Fire" and the Defender "Grass".

Output

The output is a single line of text like:

The 6XXXX type attack against the YYYYY type is ZZZZZZ

Where XXXX and YYYYY are the name of the attacker and defender types (for instance Fire and Grass in our example) and ZZZZZ is the result, based on the number and the text from the previous tables.

Example 1

Input

```
1  
3
```

Output

```
The Fire type attack against the Grass type is SUPER EFFECTIVE!
```

Example 2

Input

```
4  
5
```

Output

```
The Electric type attack against the Ground type HAS NO EFFECT.
```

Example 3

Input

```
6  
2
```

Output

```
The Rock type attack against the Water type is NOT VERY EFFECTIVE.
```


Solutions

Python

```
def main():
    # Define Pokémon types
    types = [
        "Fire", "Water", "Grass", "Electric", "Ground",
        "Rock", "Flying", "Ice", "Psychic", "Dragon"
    ]
    # Effectiveness matrix: 2.0 (Super Effective), 1.0 (Neutral), 0.5 (Not
    # Very Effective), 0.0 (No Effect)
    effectiveness = [
        # Fire
        [1.0, 0.5, 2.0, 1.0, 1.0, 0.5, 1.0, 2.0, 1.0, 0.5],
        # Water
        [2.0, 1.0, 0.5, 1.0, 2.0, 2.0, 1.0, 1.0, 1.0, 0.5],
        # Grass
        [0.5, 2.0, 1.0, 1.0, 2.0, 1.0, 0.5, 1.0, 1.0, 0.5],
        # Electric
        [1.0, 1.0, 2.0, 1.0, 0.0, 1.0, 2.0, 1.0, 1.0, 0.5],
        # Ground
        [2.0, 1.0, 0.5, 2.0, 1.0, 2.0, 0.0, 1.0, 1.0, 1.0],
        # Rock
        [2.0, 1.0, 2.0, 1.0, 0.5, 1.0, 1.0, 1.0, 1.0, 1.0],
        # Flying
        [1.0, 1.0, 2.0, 0.5, 1.0, 0.5, 1.0, 2.0, 1.0, 1.0],
        # Ice
        [1.0, 0.5, 1.0, 1.0, 1.0, 1.0, 2.0, 1.0, 1.0, 1.0],
        # Psychic
        [0.5, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0],
        # Dragon
        [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
    ]
```



<CODEWARS2026/>
LEÓN

```
[0.5, 0.5, 2.0, 1.0, 2.0, 1.0, 1.0, 0.5,
1.0, 2.0], # Ice
[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
1.0, 1.0], # Psychic
[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
1.0, 2.0] # Dragon
]

# Input: Prompt for a single line with attacker and defender indices
(e.g., "1 3")
try:
    attacker = int(input().strip())
    defender = int(input().strip())
    attacker -= 1 # Convert to 0-based index
    defender -= 1 # Convert to 0-based index
except Exception as e:
    # Handle invalid input format
    return

# Validate input
if attacker < 0 or attacker >= len(types) or defender < 0 or defender
>= len(types):
    return

# Calculate effectiveness
result = effectiveness[attacker][defender]
# Determine effectiveness description
if result == 2.0:
    effectiveness_description = "is SUPER EFFECTIVE!"
elif result == 1.0:
    effectiveness_description = "is neutral."
```



<CODEWARS2026/>
LEÓN

```
elif result == 0.5:
    effectiveness_description = "is NOT VERY EFFECTIVE."
elif result == 0.0:
    effectiveness_description = "HAS NO EFFECT."
else:
    effectiveness_description = "has an unknown effectiveness."
# Display the result
print(f"The {types[attacker]} type attack against the
{types[defender]} type {effectiveness_description}")
if __name__ == "__main__":
    main()
```

C++

```
#include <iostream>
#include <string>
#include <stdexcept>
int main() {
    // Define Pokémon types
    std::string types[] = {
        "Fire", "Water", "Grass", "Electric", "Ground",
        "Rock", "Flying", "Ice", "Psychic", "Dragon"
    };
    // Effectiveness matrix: 2.0 (Super Effective), 1.0 (Neutral), 0.5
    (Not Very Effective), 0.0 (No Effect)
    double effectiveness[10][10] = {
        // Fire    Water  Grass  Electric Ground Rock  Flying Ice
    Psychic Dragon
        {1.0,    0.5,    2.0,    1.0,    1.0,    0.5,    1.0,    2.0,
    1.0,    0.5}, // Fire
```



<CODEWARS2026/> LEÓN

```
        {2.0,    1.0,    0.5,    1.0,    2.0,    2.0,    1.0,    1.0,
1.0,    0.5}, // Water
        {0.5,    2.0,    1.0,    1.0,    2.0,    1.0,    0.5,    1.0,
1.0,    0.5}, // Grass
        {1.0,    1.0,    2.0,    1.0,    0.0,    1.0,    2.0,    1.0,
1.0,    0.5}, // Electric
        {2.0,    1.0,    0.5,    2.0,    1.0,    2.0,    0.0,    1.0,
1.0,    1.0}, // Ground
        {2.0,    1.0,    2.0,    1.0,    0.5,    1.0,    1.0,    1.0,
1.0,    1.0}, // Rock
        {1.0,    1.0,    2.0,    0.5,    1.0,    0.5,    1.0,    2.0,
1.0,    1.0}, // Flying
        {0.5,    0.5,    2.0,    1.0,    2.0,    1.0,    1.0,    0.5,
1.0,    2.0}, // Ice
        {1.0,    1.0,    1.0,    1.0,    1.0,    1.0,    1.0,    1.0,
1.0,    1.0}, // Psychic
        {1.0,    1.0,    1.0,    1.0,    1.0,    1.0,    1.0,    1.0,
1.0,    2.0} // Dragon
    };
    // Prompt for a single input
    int attacker, defender;
    try {
        std::cin >> attacker >> defender;
        attacker--; // Adjust to 0-based index
        defender--; // Adjust to 0-based index
    } catch (const std::exception& e) {
        std::cerr << "Invalid input format" << std::endl;
        return 1;
    }
```




<CODEWARS2026/>
LEÓN

```
}  
// Validate input  
if (attacker < 0 || attacker >= 10 || defender < 0 || defender >= 10)  
{  
    std::cerr << "Invalid type indices" << std::endl;  
    return 1;  
}  
// Calculate effectiveness  
double result = effectiveness[attacker][defender];  
// Determine effectiveness description  
std::string effectivenessDescription;  
if (result == 2.0) {  
    effectivenessDescription = "is SUPER EFFECTIVE!";  
} else if (result == 1.0) {  
    effectivenessDescription = "is neutral.";  
} else if (result == 0.5) {  
    effectivenessDescription = "is NOT VERY EFFECTIVE.";  
} else if (result == 0.0) {  
    effectivenessDescription = "HAS NO EFFECT.";  
} else {  
    effectivenessDescription = "has an unknown effectiveness.";  
}  
// Display the result  
std::cout << "The " << types[attacker] << " type attack against the "  
    << types[defender] << " type " << effectivenessDescription  
<< std::endl;  
    return 0;  
}
```

Java

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        // Define Pokémon types
        String[] types = {
            "Fire", "Water", "Grass", "Electric", "Ground",
            "Rock", "Flying", "Ice", "Psychic", "Dragon"
        };
        // Effectiveness matrix: 2.0 (Super Effective), 1.0 (Neutral), 0.5
        (Not Very Effective), 0.0 (No Effect)
        double[][] effectiveness = {
            // Fire    Water  Grass  Electric Ground Rock  Flying Ice
            Psychic Dragon
            {1.0,    0.5,    2.0,    1.0,    1.0,    0.5,    1.0,
            2.0,    1.0,    0.5}, // Fire
            {2.0,    1.0,    0.5,    1.0,    2.0,    2.0,    1.0,
            1.0,    1.0,    0.5}, // Water
            {0.5,    2.0,    1.0,    1.0,    2.0,    1.0,    0.5,
            1.0,    1.0,    0.5}, // Grass
            {1.0,    1.0,    2.0,    1.0,    0.0,    1.0,    2.0,
            1.0,    1.0,    0.5}, // Electric
            {2.0,    1.0,    0.5,    2.0,    1.0,    2.0,    0.0,
            1.0,    1.0,    1.0}, // Ground
            {2.0,    1.0,    2.0,    1.0,    0.5,    1.0,    1.0,
            1.0,    1.0,    1.0}, // Rock
            {1.0,    1.0,    2.0,    0.5,    1.0,    0.5,    1.0,
            2.0,    1.0,    1.0}, // Flying
```



<CODEWARS2026/>
LEÓN

```
        {0.5,    0.5,    2.0,    1.0,    2.0,    1.0,    1.0,
0.5,    1.0,    2.0}, // Ice
        {1.0,    1.0,    1.0,    1.0,    1.0,    1.0,    1.0,
1.0,    1.0,    1.0}, // Psychic
        {1.0,    1.0,    1.0,    1.0,    1.0,    1.0,    1.0,
1.0,    1.0,    2.0} // Dragon
    };
    Scanner scanner = new Scanner(System.in);
    // Prompt for a single input
    // User enters attacker and defender types in a single line (e.g.,
    "1 3")
    int attacker, defender;
    try {
        // Parse the input into attacker and defender indices
        String input = scanner.nextLine();
        attacker = Integer.parseInt(input) -1;
        String input2 = scanner.nextLine();
        defender = Integer.parseInt(input2)-1;
    } catch (Exception e) {
        // Handle invalid input format
        scanner.close();
        return;
    }
    // Validate input
    if (attacker < 0 || attacker >= types.length || defender < 0 ||
defender >= types.length) {
        scanner.close();
        return;
    }
}
```



<CODEWARS2026/>
LEÓN

```
    }  
    // Calculate effectiveness  
    double result = effectiveness[attacker][defender];  
    // Determine effectiveness description  
    String effectivenessDescription;  
    if (result == 2.0) {  
        effectivenessDescription = "is SUPER EFFECTIVE!";  
    } else if (result == 1.0) {  
        effectivenessDescription = "is neutral.";  
    } else if (result == 0.5) {  
        effectivenessDescription = "is NOT VERY EFFECTIVE.";  
    } else if (result == 0.0) {  
        effectivenessDescription = "HAS NO EFFECT.";  
    } else {  
        effectivenessDescription = "has an unknown effectiveness.";  
    }  
    // Display the result  
    System.out.println("The " + types[attacker] + " type attack  
against the "  
        + types[defender] + " type " + effectivenessDescription);  
    scanner.close();  
}  
}
```


7

The Mysterious Pascal's Triangle

5 points

Introduction

Welcome, young mathematician! Deep in Numeroland, the land of Numbers, the wise wizard Pascal has hidden a magical triangle. This triangle holds secrets of patterns and powers, but only those who can decode its last row will unlock its mysteries!



The wizard Pascal didn't just make up the numbers in his triangle, he used a clever spell! Here's how you can conjure up the numbers yourself:

- The very top of the triangle always starts with a single 1. This is the tip of the wizard's hat!
- Every row starts and ends with 1. These are the magical borders that protect the triangle.
- Every other number inside the triangle is created by adding the two numbers directly above it—one from the left and one from the right. It's like each number is the child of two parents above it!

For example, to create the triangle of size 2:

```
1
1 1
1 2 1
```

See that 2 in the middle? It's made by adding the two 1s above it ($1 + 1 = 2$). Every number in the triangle (except the edges) is made this way!

Exercise

Your quest:

1. Input: The wizard will give you a number, N ($1 \leq N \leq 50$). This is the number of rows in Pascal's Triangle he wants you to explore.
2. Triangle Clues: For the first N rows (except the last), draw the triangle using only / and \ characters, so the triangle looks magical and mysterious.
3. The Final Secret: The last row of Pascal's Triangle is the key! Print each number in the last row, but with a twist: write each digit vertically, one below the other. For example, if you need to write 10, the "1" will be in the first line and the "0" in the second line. Are you ready to reveal the secrets of Pascal's Triangle and become a legend in the land of Numbers? Good luck, code wizard!

Input

The size of the Pascal's triangle with a single integer number lower than 50.

Output

If the number is less than 1, shout:

```
Error! Triangulo imposible.
```

If the number is greater than 50, warn:

```
Error! Triangulo demasiado grande.
```

The triangle of the specified size with the last row values in vertical format. Please note that the numbers for the last row values are padded with spaces to the right to have the same width

Important Note: Please Note that when a triangle is drawn, a blank line is added behind the triangle. This line is a white spaces line, as long as the previous line.

Example 1

Input

3

Output

```
  /\n /\n /\n /\n1 3 3 1
```

Example 2

Input

5

Output

```
  /\n /\n /\n /\n /\n /\n1 5 1 1 5 1
  0 0
```

Note: the last row for 5 is "1 5 10 10 5 1", so the digits of "10" are printed vertically.

Solutions

Python

```
def main():
    try:
        numRows = int(input())
    except Exception:
        print("Error! Triangulo imposible.")
        return

    if numRows < 1:
        print("Error! Triangulo imposible.")
        return

    if numRows > 50:
        print("Error! Triangulo demasiado grande.")
        return

    # Calculate only the last line of Pascal's Triangle, using Python's
    big integers
    row = [1] * (numRows + 1)
    for i in range(1, numRows):
        for j in range(i, 0, -1):
            row[j] += row[j - 1]

    # Print the n-1 first lines of Pascal's Triangle as '/' and '\',
    centered
    for i in range(numRows):
        print(' ' * (numRows - i - 1), end='')
        print(''.join('/') if j % 2 == 0 else '\\' for j in range(i * 2 +
2)))
```




<CODEWARS2026/>
LEÓN

```
# Print the last line, each digit vertically, with spaces between
numbers
num_strs = [str(n) for n in row]
digit_pos = 0
while True:
    has_more_digits = False
    line = []
    for num_str in num_strs:
        if digit_pos < len(num_str):
            line.append(num_str[digit_pos])
            has_more_digits = True
        else:
            line.append(' ')
    print(' '.join(line))
    if not has_more_digits:
        break
    digit_pos += 1
if __name__ == "__main__":
    main()
```

C++

```
// PascalsTriangle.cpp : This file contains the 'main' function. Program
execution begins and ends there.
//
#include <iostream>
#include <vector>
#include <string>
int main()
{
```



<CODEWARS2026/>
LEÓN

```
// We need to get an integer from the standard input
int numRows;
std::cin >> numRows;

// If the number is less than 1, we should show the message "Error!
Triangulo imposible."
if (numRows < 1) {
    std::cout << "Error! Triangulo imposible." << std::endl;
    return 0;
}

// If the number is greater than 50, we should show the message
"Error! Triangulo demasiado grande."
if (numRows > 50) {
    std::cout << "Error! Triangulo demasiado grande." << std::endl;
    return 0;
}

// We should calculate only the last line of Pascal's Triangle, using
64bit integers
std::vector<int64_t> row(numRows + 1, 1);
for (int i = 1; i < numRows; ++i) {
    for (int j = i; j > 0; --j) {
        row[j] += row[j - 1];
    }
}

// Print the n-1 first lines of Pascal's Triangle only as characters
'/' and '\' indicating the relationship of the values, centered along the
base
for (int i = 0; i < numRows; ++i) {
    // Print leading spaces
```



```
for (int s = 0; s < numRows - i - 1; ++s) {
    std::cout << ' ';
}
// Print the values with '/' and '\\'
for (int j = 0; j <= i*2 + 1; ++j) {
    std::cout << (j % 2 == 0 ? '/' : '\\') ;
}
std::cout << '\n';
}

// Print the last line of Pascal's Triangle. Each number should be
written in vertical, that is, each digit in it's own line whith one space
between different numbers.
for (int digitPos = 0; ; ++digitPos) {
    bool hasMoreDigits = false;
    for (int j = 0; j <= numRows; ++j) {
        std::string numStr = std::to_string(row[j]);
        if (digitPos < numStr.size()) {
            std::cout << numStr[digitPos];
            hasMoreDigits = true;
        } else {
            std::cout << ' ';
        }
        if (j < numRows) {
            std::cout << ' '; // space between numbers
        }
    }
    std::cout << '\n';
    if (!hasMoreDigits) {
```



<CODEWARS2026/> LEÓN

```
        break;
    }
}
}
```

Java

```
import java.util.Scanner;
public class PascalsTriangle {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int numRows;
        try {
            numRows = Integer.parseInt(sc.nextLine().trim());
        } catch (Exception e) {
            System.out.println("Error! Triangulo imposible.");
            return;
        }
        if (numRows < 1) {
            System.out.println("Error! Triangulo imposible.");
            return;
        }
        if (numRows > 50) {
            System.out.println("Error! Triangulo demasiado grande.");
            return;
        }
        // Calcular solo la última fila del triángulo de Pascal
        java.math.BigInteger[] row = new java.math.BigInteger[numRows +
1];
```




<CODEWARS2026/>
LEÓN

```
        for (int i = 0; i <= numRows; i++) row[i] =
java.math.BigInteger.ONE;
        for (int i = 1; i < numRows; i++) {
            for (int j = i; j > 0; j--) {
                row[j] = row[j].add(row[j - 1]);
            }
        }
        // Imprimir las n-1 primeras líneas con '/' y '\', centradas
        for (int i = 0; i < numRows; i++) {
            for (int s = 0; s < numRows - i - 1; s++) System.out.print("
");
            for (int j = 0; j < i * 2 + 2; j++) System.out.print(j % 2 ==
0 ? "/" : "\\");
            System.out.println();
        }
        // Imprimir la última línea, cada dígito verticalmente, con
espacios entre números
        String[] numStrs = new String[row.length];
        int maxLen = 0;
        for (int i = 0; i < row.length; i++) {
            numStrs[i] = row[i].toString();
            if (numStrs[i].length() > maxLen) maxLen =
numStrs[i].length();
        }
        int digitPos = 0;
        while (true) {
            boolean hasMoreDigits = false;
            StringBuilder line = new StringBuilder();
```



<CODEWARS2026/>
LEÓN

```
for (int i = 0; i < numStrs.length; i++) {  
    if (digitPos < numStrs[i].length()) {  
        line.append(numStrs[i].charAt(digitPos));  
        hasMoreDigits = true;  
    } else {  
        line.append(" ");  
    }  
    if (i < numStrs.length - 1) line.append(" ");  
}  
System.out.println(line);  
if (!hasMoreDigits) break;  
digitPos++;  
}  
}
```

<let's code/>



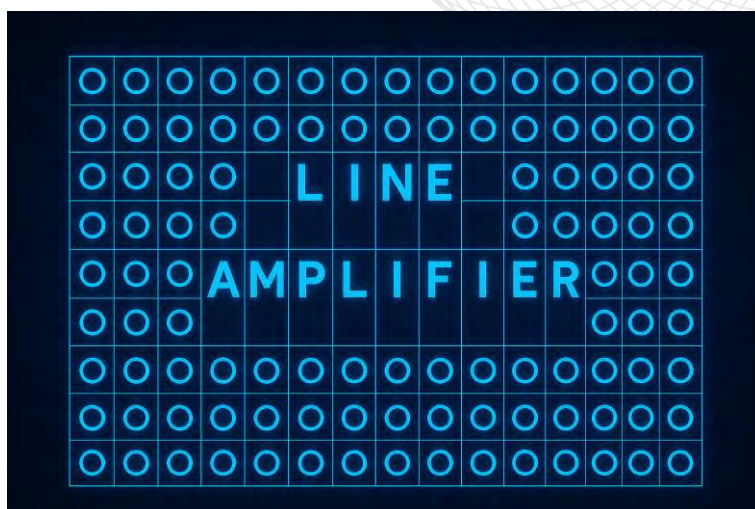


<CODEWARS2026/>
LEÓN

8

Line Amplifier

6 points



Introduction

You are requested to write a program that amplifies lines drawn through standard input with letters '8'. The program will receive a simulated 10x10 board thorough the standard input where different lines could be drawn suing the character 'X'. For instance:

```
X
X
X
XXXX
```

In this case we have drawn letter 'L'. Note that 2 empty lines and 3 empty columns exist before the first 'X'.

You are asked to make kind of a "Bold" function so that the 'X's are surrounded by letters 'O'.

For instance with the previous input, the expected output is:

```
OOO
OXO
OXO
OXOOOO
OXXXXO
OOOOOO
```

Note that the board is still 10x10 (10 lines and 10 columns).

Exercise

You are asked to write a program that amplifies anything that's written on the board. To make things easier, the drawings will only have 90° angles (nothing like the following):



Input

The program will read 10 lines (which will be the board). Note that there will always be at least one blank line and one blank column before the first 'X'. And an empty line and column after the last 'X'. This is to give space for the amplification

Output

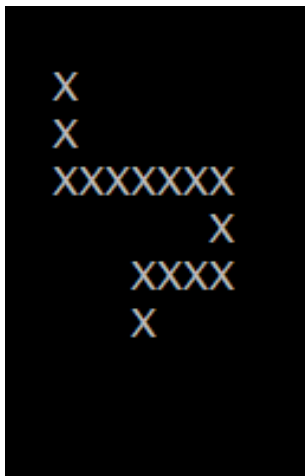
The program will write a 10 dash separator, and then the 10 lines board with the amplified drawing.

Example 1

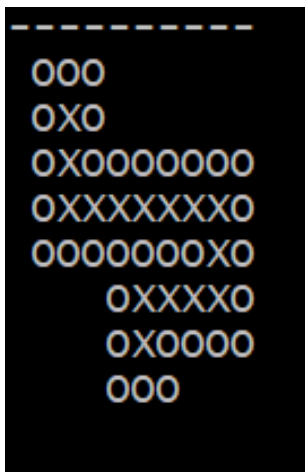
Input



<CODEWARS2026/> LEÓN



Output

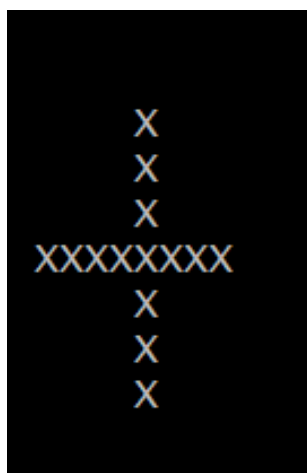


Example 2

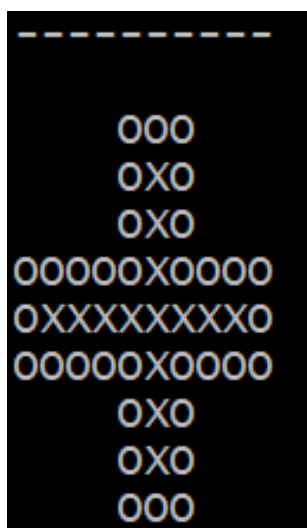
Input



<CODEWARS2026/>
LEÓN



Output



Solutions

Python

```
def ampliar_dibujo():  
    # Leer 10 líneas desde entrada estándar
```



<CODEWARS2026/> LEÓN

```
lineas = [input() for _ in range(10)]
print ("-----")
# Determinar dimensiones originales
filas = len(lineas)
columnas = max(len(linea) for linea in lineas)
# Crear matriz original con espacios
matriz = [list(linea.ljust(columnas)) for linea in lineas]
# Crear nueva matriz con borde adicional
nueva_filas = filas
nueva_columnas = columnas +1
nueva_matriz = [[' ' for _ in range(nueva_columnas)] for _ in
range(nueva_filas)]
# Copiar las X en la posición desplazada
for i in range(filas):
    for j in range(columnas):
        if matriz[i][j] == 'X':
            nueva_matriz[i][j] = 'X'
# Añadir 0 alrededor del bloque que contiene X
for i in range(nueva_filas):
    for j in range(nueva_columnas):
        if nueva_matriz[i][j] == 'X':
            # Marcar las posiciones vecinas como 0 si están vacías
            for di in [-1, 0, 1]:
                for dj in [-1, 0, 1]:
                    ni, nj = i + di, j + dj
                    if 0 <= ni < nueva_filas and 0 <= nj <
nueva_columnas:
                        if nueva_matriz[ni][nj] == ' ':
```




<CODEWARS2026/>
LEÓN

```
nueva_matriz[ni][nj] = '0'

# Imprimir el resultado
for fila in nueva_matriz:
    print(''.join(fila).rstrip())
# Ejecutar la función
ampliar_dibujo()

C++
#include <iostream>
#include <vector>
#include <string>
#include <algorithm>
using namespace std;
int main() {
    vector<string> lineas(10);
    for (int i = 0; i < 10; i++) {
        getline(cin, lineas[i]);
    }
    cout << "-----" << endl;
    int filas = lineas.size();
    int columnas = 0;
    for (auto &linea : lineas) {
        columnas = max(columnas, (int)linea.size());
    }
    // Crear matriz original
    vector<vector<char>> matriz(filas, vector<char>(columnas, ' '));
    for (int i = 0; i < filas; i++) {
        for (int j = 0; j < (int)lineas[i].size(); j++) {
            matriz[i][j] = lineas[i][j];
        }
    }
}
```



<CODEWARS2026/>
LEÓN

```
    }  
}  
// Nueva matriz con borde adicional  
int nuevaFilas = filas;  
int nuevaColumnas = columnas + 1;  
vector<vector<char>> nuevaMatriz(nuevaFilas,  
vector<char>(nuevaColumnas, ' '));  
// Copiar las X  
for (int i = 0; i < filas; i++) {  
    for (int j = 0; j < columnas; j++) {  
        if (matriz[i][j] == 'X') {  
            nuevaMatriz[i][j] = 'X';  
        }  
    }  
}  
// Añadir 0 alrededor  
for (int i = 0; i < nuevaFilas; i++) {  
    for (int j = 0; j < nuevaColumnas; j++) {  
        if (nuevaMatriz[i][j] == 'X') {  
            for (int di = -1; di <= 1; di++) {  
                for (int dj = -1; dj <= 1; dj++) {  
                    int ni = i + di, nj = j + dj;  
                    if (ni >= 0 && ni < nuevaFilas && nj >= 0 && nj <  
nuevaColumnas) {  
                        if (nuevaMatriz[ni][nj] == ' ') {  
                            nuevaMatriz[ni][nj] = '0';  
                        }  
                    }  
                }  
            }  
        }  
    }  
}
```



<CODEWARS2026/>
LEÓN

```
        }
    }
}

// Imprimir resultado
for (int i = 0; i < nuevaFilas; i++) {
    string fila(nuevaMatriz[i].begin(), nuevaMatriz[i].end());
    // Eliminar espacios a la derecha
    fila.erase(find_if(fila.rbegin(), fila.rend(), [](unsigned char
ch) {
        return ch != ' ';
    }).base(), fila.end());
    cout << fila << endl;
}
return 0;
}
```

Java

```
import java.util.*;

public class Line_Amplifier {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String[] lineas = new String[10];
        // Leer 10 líneas
        for (int i = 0; i < 10; i++) {
            lineas[i] = sc.nextLine();
        }
        System.out.println("-----");
    }
}
```

<let's code/>





<CODEWARS2026/>
LEÓN

```
int filas = lineas.length;
int columnas = 0;
for (String linea : lineas) {
    columnas = Math.max(columnas, linea.length());
}
// Crear matriz original
char[][] matriz = new char[filas][columnas];
for (int i = 0; i < filas; i++) {
    String linea = lineas[i];
    for (int j = 0; j < columnas; j++) {
        matriz[i][j] = (j < linea.length()) ? linea.charAt(j) : '
';
    }
}
// Nueva matriz con borde adicional
int nuevaFilas = filas;
int nuevaColumnas = columnas + 1;
char[][] nuevaMatriz = new char[nuevaFilas][nuevaColumnas];
for (int i = 0; i < nuevaFilas; i++) {
    Arrays.fill(nuevaMatriz[i], ' ');
}
// Copiar las X
for (int i = 0; i < filas; i++) {
    for (int j = 0; j < columnas; j++) {
        if (matriz[i][j] == 'X') {
            nuevaMatriz[i][j] = 'X';
        }
    }
}
```




<CODEWARS2026/>
LEÓN

```
    }  
    // Añadir 0 alrededor  
    for (int i = 0; i < nuevaFilas; i++) {  
        for (int j = 0; j < nuevaColumnas; j++) {  
            if (nuevaMatriz[i][j] == 'X') {  
                for (int di = -1; di <= 1; di++) {  
                    for (int dj = -1; dj <= 1; dj++) {  
                        int ni = i + di, nj = j + dj;  
                        if (ni >= 0 && ni < nuevaFilas && nj >= 0 &&  
nj < nuevaColumnas) {  
                            if (nuevaMatriz[ni][nj] == ' ') {  
                                nuevaMatriz[ni][nj] = '0';  
                            }  
                        }  
                    }  
                }  
            }  
        }  
    }  
    // Imprimir resultado  
    for (int i = 0; i < nuevaFilas; i++) {  
        System.out.println(new  
String(nuevaMatriz[i]).stripTrailing());  
    }  
}
```

9

Tie Break

6 points

Introduction

You work as a programmer for an IT company, and the Roland Garros tennis tournament has commissioned your team to develop a project for an automatic scoreboard that doesn't require manual intervention.

Exercise

One person in your company is responsible for the visual design, another ensures that the AI detects who wins each point, another handles the point counting during the set, and you've been asked to create a program that counts the points during tie-breaks.



As is well known, the scoring system in tennis matches is peculiar: players score 15, 30, 40 points... But in tie-breaks, the counting is normal: 1, 2, 3, 4... So you breathe a sigh of relief.

But then you discover that tie-breaks also have their own rules: The tie-break is won by the first player to reach 7 points with a 2-point lead. This means that if the score is 7-6, the match isn't over yet and will continue to 8. If the player with 6 wins the next point, the score becomes 7-7 and will extend to 9... In a tie-break, the score could easily end up being 19-17 without any problem.

Input

As it's well-known computers only work with 0's and 1's. The AI will therefore only return a list of numbers (0 or 1) representing which player won each point (0 means that player 1 won the point, while 1 means that player 2 won the point). An example input might be:

```
0,1,0,0,1,1,1,0,0,1,0,0
```

Output

The program will generate an output in the following format (in spanish):

```
Jugador <GANADOR> gana el tie-break <RESULTADO>
```

For example, for the given input, the output would be:

```
Jugador 1 gana el tie-break 7-5
```

It could happen that the tie-break has not finished yet. In that case, the output will look like this:

```
El tie-break aún no ha terminado. Van 4-3
```

In this case, the first number corresponds to Player 1's points and the second to Player 2's points.

Example1

Input

```
0,1,0,0,1,1,1,0,0,1,0,0
```

Output

Jugador 1 gana el tie-break 7-5

Example2

Input

0,1,0,0,1,1,1,0,0,1,0,1,0,1,0,0,1,1

Output

Jugador 1 gana el tie-break 10-8

Solutions

Python

```
def tie_break_result(puntos):
    jugador1 = jugador2 = 0
    for punto in puntos:
        if punto == 0:
            jugador1 += 1
        elif punto == 1:
            jugador2 += 1
    # Comprobar si alguien ha ganado el tie-break
    if jugador1 >= 7 and jugador1 - jugador2 >= 2:
        return f"Jugador 1 gana el tie-break {jugador1}-{jugador2}"
    if jugador2 >= 7 and jugador2 - jugador1 >= 2:
        return f"Jugador 2 gana el tie-break {jugador2}-{jugador1}"
    # Si nadie ha ganado aún
    return f"El tie-break aún no ha terminado. Van {jugador1}-{jugador2}"

entrada = input("")
puntos = [int(x) for x in entrada.split(",")]
```




<CODEWARS2026/>
LEÓN

```
print(tie_break_result(puntos))
```

C++

```
#include <iostream>
#include <sstream>
#include <vector>
#include <string>
using namespace std;
string tieBreakResult(const vector<int>& puntos) {
    int jugador1 = 0, jugador2 = 0;
    for (int punto : puntos) {
        if (punto == 0) {
            jugador1++;
        } else if (punto == 1) {
            jugador2++;
        }
        // Comprobar si alguien ha ganado el tie-break
        if (jugador1 >= 7 && jugador1 - jugador2 >= 2) {
            return "Jugador 1 gana el tie-break " + to_string(jugador1) +
                "-" + to_string(jugador2);
        }
        if (jugador2 >= 7 && jugador2 - jugador1 >= 2) {
            return "Jugador 2 gana el tie-break " + to_string(jugador2) +
                "-" + to_string(jugador1);
        }
    }
    return "El tie-break aún no ha terminado. Van " + to_string(jugador1)
        + "-" + to_string(jugador2);
}
```

<let's code/>





<CODEWARS2026/>
LEÓN

```
int main() {
    string entrada;
    getline(cin, entrada);
    stringstream ss(entrada);
    string token;
    vector<int> puntos;
    while (getline(ss, token, ',')) {
        puntos.push_back(stoi(token));
    }
    cout << tieBreakResult(puntos) << endl;
    return 0;
}
```

Java

```
import java.util.Scanner;

public class Tiebreak {
    public static String tieBreakResult(int[] puntos) {
        int jugador1 = 0, jugador2 = 0;
        for (int punto : puntos) {
            if (punto == 0) {
                jugador1++;
            } else if (punto == 1) {
                jugador2++;
            }
        }
        // Comprobar si alguien ha ganado el tie-break
        if (jugador1 >= 7 && jugador1 - jugador2 >= 2) {
            return "Jugador 1 gana el tie-break " + jugador1 + "-" +
jugador2;
        }
    }
}
```

<let's code/>



```
        if (jugador2 >= 7 && jugador2 - jugador1 >= 2) {  
            return "Jugador 2 gana el tie-break " + jugador2 + "-" +  
jugador1;  
        }  
    }  
    return "El tie-break aún no ha terminado. Van " + jugador1 + "-" +  
jugador2;  
}  
public static void main(String[] args) {  
    Scanner sc = new Scanner(System.in);  
    String entrada = sc.nextLine();  
    String[] partes = entrada.split(",");  
    int[] puntos = new int[partes.length];  
    for (int i = 0; i < partes.length; i++) {  
        puntos[i] = Integer.parseInt(partes[i].trim());  
    }  
    System.out.println(tieBreakResult(puntos));  
}  
}
```

10

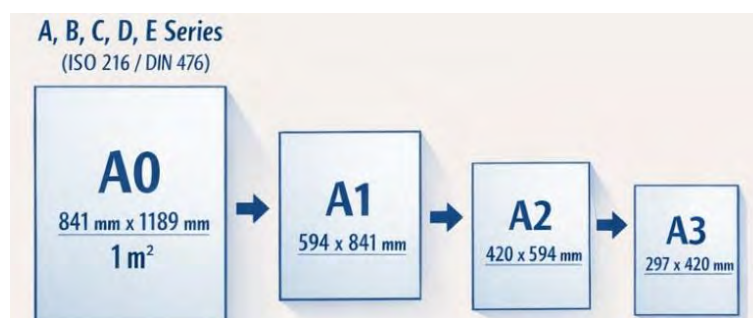
The ISO 216

7 points

Introduction

The printing paper format has standardized dimensions, which helps to unify production and printing criteria, and at the same time serves as a guide in design, graphic arts in design, graphic arts, advertising, press, etc. There are two main classifications of formats:

- European format, widely used in many countries around the world, includes the A, B, C, D and E series defined in ISO 216, which is based on the DIN 476 standard defined in 1922.
- American format, mainly used in the United States, Canada and Mexico, includes traditional formats such as letter, legal and tabloid.



The purpose behind the standardization of formats is to maximize paper usage to make the best possible use of it and minimize waste.

The ISO 216 standard defines paper formats whose sides maintain a constant ratio, such that when a sheet is divided in half by its length, each half still has the same ratio between its sides as the original sheet.

As a summary, for ISO/DIN formats:

- The ratio between their sides (x/y) is constant.
- The area and the sides lengths are expressed in square millimeters (mm^2) and millimeters (mm), respectively, and always rounded down to the nearest integer.

Exercise

You are asked to develop a program that, starting from the ISO A series whose base format is A0, which has a surface of 1 m^2 and dimensions of $841 \times 1189 \text{ mm}$, is able to calculate the approximate area of any successive format (A1, A2, A3, A4, A5, A6, A7 or A8).

These derived formats, whose number indicates the number of times the original sheet has been cut in half, have an area equal to half that of the previous format. Thus, an A1 sheet has an area of $\frac{1}{2} \text{ m}^2$, an A2 sheet has an area of $\frac{1}{4} \text{ m}^2$, and so on.

In addition to calculating the surface area of the specified format in mm^2 , the program must also provide the length of each side in millimeters (mm). In addition, It must also verify whether the input corresponds to a valid A format; otherwise, the following message must be displayed:

The entry does not correspond to a format A.

Note: All values should be rounded down to the nearest whole number.

Note 2: There is a simple relationship between the dimensions of successive formats. If a sheet of format A1 has dimensions $x \times y$ (with y being the shorter side), then a sheet of format A2 will have dimensions $y \times z$, where y remains the same value and becomes the longer side.

Input

A single line like A1, A2, A3, A4, A5...

Output

A single line with the following text:

Superficie: <area>; Lado mayor: <longer side>; Lado menor: <shorter side>

Example 1

Input

A1

Output

Superficie: 500000; Lado mayor: 841; Lado menor: 594

Example 2

Input

A5

Output

Superficie: 31250; Lado mayor: 210; Lado menor: 148

Solutions

Python

```
def calculate_format_area_and_dimensions(format_str: str):  
    # Validate input format  
    if not format_str.startswith('A') or not format_str[1:].isdigit():  
        print("The entry does not correspond to a format A.")  
    return
```



```
# Extract the number from the format (e.g., 'A1' -> 1)
n = int(format_str[1:])
# Ensure the number is between 0 and 8 (A0 to A8)
if n < 0 or n > 8:
    print("The entry does not correspond to a format A.")
    return
# Initial dimensions and area for A0
width = 841
height = 1189
area = 1000000 # in mm² (1 m² = 1000000 mm²)
# Calculate the format's area and dimensions based on A0
for i in range(n):
    # The shorter side of the previous sheet becomes the longer side
of the next
    # The longer side is halved to become the new shorter side
    height, width = width, height // 2
    area = area // 2 # Halve the area
# Output the results
print(f"Superficie: {area}; Lado mayor: {height}; Lado menor:
{width}")
# Main function to get the input and call the calculation function
def main():
    input_format = input().strip() # Read the input
    calculate_format_area_and_dimensions(input_format)
# Run the program
if __name__ == "__main__":
    main()
```

C++



<CODEWARS2026/>
LEÓN

```
#include <iostream>
#include <string>
#include <cstdlib> // For std::stoi
using namespace std;
// Function to calculate format area and dimensions
void calculateFormatAreaAndDimensions(const string& format_str) {
    // Validate input format
    if (format_str.empty() || format_str[0] != 'A' ||
format_str.substr(1).find_first_not_of("0123456789") != string::npos) {
        cout << "The entry does not correspond to a format A." << endl;
        return;
    }
    // Extract the number from the format (e.g., 'A1' -> 1)
    int n;
    try {
        n = stoi(format_str.substr(1)); // Convert the substring to
integer
    } catch (const invalid_argument&) {
        cout << "The entry does not correspond to a format A." << endl;
        return;
    }
    // Ensure the number is between 0 and 8 (A0 to A8)
    if (n < 0 || n > 8) {
        cout << "The entry does not correspond to a format A." << endl;
        return;
    }
    // Initial dimensions and area for A0
    int width = 841;
```




<CODEWARS2026/>
LEÓN

```
int height = 1189;
int area = 1000000; // in mm² (1 m² = 1000000 mm²)
// Calculate the format's area and dimensions based on A0
for (int i = 0; i < n; ++i) {
    // For each step, we swap dimensions and halve the larger side
    if (width > height) {
        swap(width, height);
    }
    height = height / 2; // Halve the height
    area = area / 2; // Halve the area
}
// Output the results
cout << "Superficie: " << area << "; Lado mayor: " << max(width,
height) << "; Lado menor: " << min(width, height) << endl;
}
int main() {
    string input_format;
    getline(cin, input_format); // Read the input
    calculateFormatAreaAndDimensions(input_format); // Call the
calculation function
    return 0;
}
```

Java

```
import java.util.Scanner;
public class FormatDimensions {
    // Function to calculate format area and dimensions
    public static void calculateFormatAreaAndDimensions(String formatStr)
{
```

```
// Validate input format
if (formatStr.isEmpty() || formatStr.charAt(0) != 'A' ||
!formatStr.substring(1).matches("\\d+")) {
    System.out.println("The entry does not correspond to a format
A.");
    return;
}
// Extract the number from the format (e.g., 'A1' -> 1)
int n;
try {
    n = Integer.parseInt(formatStr.substring(1)); // Convert the
substring to integer
} catch (NumberFormatException e) {
    System.out.println("The entry does not correspond to a format
A.");
    return;
}
// Ensure the number is between 0 and 8 (A0 to A8)
if (n < 0 || n > 8) {
    System.out.println("The entry does not correspond to a format
A.");
    return;
}
// Initial dimensions and area for A0
int width = 841;
int height = 1189;
int area = 1000000; // in mm² (1 m² = 1000000 mm²)
// Calculate the format's area and dimensions based on A0
```



<CODEWARS2026/>
LEÓN

```
for (int i = 0; i < n; i++) {  
    // For each step, we swap dimensions and halve the larger side  
    if (width > height) {  
        int temp = width;  
        width = height;  
        height = temp;  
    }  
    height = height / 2; // Halve the height  
    area = area / 2; // Halve the area  
}  
// Output the results  
System.out.printf("Superficie: %d; Lado mayor: %d; Lado menor:  
%d\n", area, Math.max(width, height), Math.min(width, height));  
}  
public static void main(String[] args) {  
    Scanner scanner = new Scanner(System.in);  
    // Read the input  
    String inputFormat = scanner.nextLine().trim(); // Read the input  
    format and trim any extra spaces  
    // Call the calculation function  
    calculateFormatAreaAndDimensions(inputFormat);  
}  
}
```

<let's code/>



11

Rock, Paper, Scissors

7 points

Introduction

Son Goku and Vegeta, two of the most powerful Saiyan warriors, are preparing for the ultimate battle against Perfect Cell in his self-declared Martial Arts Tournament. However, there's only one spot for the next challenger, and Son Goku and Vegeta can't decide who should fight first.



To settle this, they decide to play a best of three "Rock, Paper, Scissors" game. You are Son Goku and the computer will be Vegeta. You must compete against the other computer in this decisive game. The battle continues until you or the computer wins at least 2 games (Best of 3 possible winning results are: (1-2), (2-1), (2-0), (0-2)).

Exercise

Between you and me, Vegeta is quite foolish: it will always reply in a specific order. In the first turn, it will always show "Scissors", in the second turn it will show "Rock" and in the third turn it will always show "Paper", and then in the fourth turn it will show again "Scissors", and in the fifth it will show "Rock" again, and so on.

You are asked to write a program that read the Son Goku selections and play against Vegeta (computer) until one of the wins (or the game end in a draw). The read selections will play against the computer's selections that will always be the same: SRPSRPSRPSRP...

The program will simply write the name of the Winner or if it's been a draw (See explanations below).

Input

The program will read a single line with letters P, R or S, that specific the decisions (Paper,Rock,Scissors) that Son Goku would take in possible future. For instance:

```
RPSRPSRPSRPSRPSRPSR
```

Output

The program will simply write the name of the Winner:

```
SON GOKU WILL FIGHT
```

or

```
VEGETA WILL FIGHT
```

or

```
IT'S A DRAW; PLAY AGAIN
```

Example 1

Input

RPSRPSRPSRPSRPSRPSR

Output

SON GOKU WILL FIGHT

Example 2

Input

SRPSRPSRP

Output

IT'S A DRAW; PLAY AGAIN

Example 3

Input

SRPPSP

Output

VEGETA WILL FIGHT

Solutions

Python

```
def determine_winner(son_goku_move, vegeta_move):  
    # Determine the winner of one round  
    if son_goku_move == vegeta_move:  
        return 'draw' # It's a draw if both moves are the same  
    elif (son_goku_move == 'R' and vegeta_move == 'S') or \  
        (son_goku_move == 'S' and vegeta_move == 'P') or \  
        (son_goku_move == 'P' and vegeta_move == 'R'):
```



<CODEWARS2026/> LEÓN

```
        return 'son_goku' # Son Goku wins
    else:
        return 'vegeta' # Vegeta wins
def main():
    # Read the input string (Son Goku's sequence of moves)
    son_goku_moves = input().strip()
    # Vegeta's moves cycle: Scissors -> Rock -> Paper -> Scissors -> Rock
    -> Paper ...
    vegeta_moves = ['S', 'R', 'P']
    # Initialize counters for wins
    son_goku_wins = 0
    vegeta_wins = 0
    round_count = 0
    # We only need to play until one player wins 2 rounds
    for i in range(len(son_goku_moves)):
        son_goku_move = son_goku_moves[i]
        vegeta_move = vegeta_moves[i % 3] # Cycle through Vegeta's moves
        #print(son_goku_move, " VS ", vegeta_move)
        # Determine the winner of this round
        result = determine_winner(son_goku_move, vegeta_move)
        if result == 'son_goku':
            son_goku_wins += 1
        elif result == 'vegeta':
            vegeta_wins += 1
        #print(son_goku_wins, " VS ", vegeta_wins)
        # Check if any player has won 2 rounds
        if son_goku_wins == 2:
            print("SON GOKU WILL FIGHT")
```



<CODEWARS2026/>
LEÓN

```
        return
    elif vegeta_wins == 2:
        print("VEGETA WILL FIGHT")
        return
    round_count += 1
    # If we finish all rounds without a 2-0 decision, it's a draw
    print("IT'S A DRAW; PLAY AGAIN")
# Execute the main function
main()
```

C++

```
#include <iostream>
#include <string>
using namespace std;
string determine_winner(char son_goku_move, char vegeta_move) {
    // Determine the winner of one round
    if (son_goku_move == vegeta_move) {
        return "draw"; // It's a draw if both moves are the same
    } else if ((son_goku_move == 'R' && vegeta_move == 'S') ||
               (son_goku_move == 'S' && vegeta_move == 'P') ||
               (son_goku_move == 'P' && vegeta_move == 'R')) {
        return "son_goku"; // Son Goku wins
    } else {
        return "vegeta"; // Vegeta wins
    }
}
int main() {
    // Read the input string (Son Goku's sequence of moves)
    string son_goku_moves;
```




<CODEWARS2026/>
LEÓN

```
cin >> son_goku_moves;
// Vegeta's moves cycle: Scissors -> Rock -> Paper -> Scissors -> Rock
-> Paper ...
char vegeta_moves[] = {'S', 'R', 'P'};
int son_goku_wins = 0;
int vegeta_wins = 0;
// Play the game until one player wins 2 rounds
for (int i = 0; i < son_goku_moves.length(); i++) {
    char son_goku_move = son_goku_moves[i];
    char vegeta_move = vegeta_moves[i % 3]; // Cycle through Vegeta's
moves
    // Determine the winner of this round
    string result = determine_winner(son_goku_move, vegeta_move);
    if (result == "son_goku") {
        son_goku_wins++;
    } else if (result == "vegeta") {
        vegeta_wins++;
    }
    // Check if any player has won 2 rounds
    if (son_goku_wins == 2) {
        cout << "SON GOKU WILL FIGHT" << endl;
        return 0;
    } else if (vegeta_wins == 2) {
        cout << "VEGETA WILL FIGHT" << endl;
        return 0;
    }
}
// If we finish all rounds without a 2-0 decision, it's a draw
```



```
    cout << "IT'S A DRAW; PLAY AGAIN" << endl;  
    return 0;  
}
```

Java

```
import java.util.Scanner;  
  
public class RockPaperScissors {  
    // Determine the winner of one round  
    public static String determineWinner(char sonGokuMove, char  
vegetaMove) {  
        if (sonGokuMove == vegetaMove) {  
            return "draw"; // It's a draw if both moves are the same  
        } else if ((sonGokuMove == 'R' && vegetaMove == 'S') ||  
                    (sonGokuMove == 'S' && vegetaMove == 'P') ||  
                    (sonGokuMove == 'P' && vegetaMove == 'R')) {  
            return "son_goku"; // Son Goku wins  
        } else {  
            return "vegeta"; // Vegeta wins  
        }  
    }  
  
    public static void main(String[] args) {  
        // Read the input string (Son Goku's sequence of moves)  
        Scanner sc = new Scanner(System.in);  
        String sonGokuMoves = sc.nextLine();  
        sc.close();  
        // Vegeta's moves cycle: Scissors -> Rock -> Paper -> Scissors ->  
Rock -> Paper ...  
        char[] vegetaMoves = {'S', 'R', 'P'};  
        int sonGokuWins = 0;
```

```
int vegetaWins = 0;
// Play the game until one player wins 2 rounds
for (int i = 0; i < sonGokuMoves.length(); i++) {
    char sonGokuMove = sonGokuMoves.charAt(i);
    char vegetaMove = vegetaMoves[i % 3]; // Cycle through
Vegeta's moves
    // Determine the winner of this round
    String result = determineWinner(sonGokuMove, vegetaMove);
    if (result.equals("son_goku")) {
        sonGokuWins++;
    } else if (result.equals("vegeta")) {
        vegetaWins++;
    }
    // Check if any player has won 2 rounds
    if (sonGokuWins == 2) {
        System.out.println("SON GOKU WILL FIGHT");
        return;
    } else if (vegetaWins == 2) {
        System.out.println("VEGETA WILL FIGHT");
        return;
    }
}
// If we finish all rounds without a 2-0 decision, it's a draw
System.out.println("IT'S A DRAW; PLAY AGAIN");
}
}
```


12

The time-traveling gunslinger

8 points

Introduction

In a distant past, a gunslinger named Rex, the Time Traveler, has been assigned a very special mission. He is known not only for his skill with weapons, but also for his ability to travel through time and space. His mission is crucial: to ensure that historical events happen as they should, without alterations.



To complete his task, Rex must travel to different eras and places around the world, and he often encounters an unexpected difficulty: date formats vary depending on the time period and the location.

For example, in the future, dates are written in the format YYYY-MM-DD, while in the past, some cultures write dates in the format DD/MM/YYYY, and in ancient civilizations, dates may be represented in entirely different ways.

Rex needs your help to easily convert dates from one format to another. You will be his trusted programmer, responsible for writing the algorithm that allows him to switch between date formats depending on the era he is in.

Exercise

Your task is to create a program that receives a date in a specific format and converts it to another format that Rex needs to complete his mission. The program must read the date in the input format, validate it, and then convert it to the output format.

The date formats Rex needs to handle are:

```
"YYYY-MM-DD" (e.g., "2024-12-10")  
"DD/MM/YYYY" (e.g., "10/12/2024")
```

Input

The program will receive a single line of text in the format:

```
source_date_format --> destination_date_format --> date, where:
```

where:

- source_date_format can be "YYYY-MM-DD" or "DD/MM/YYYY".
- destination_date_format will also be one of these two formats.
- date is the date to be converted.

```
YYYY-MM-DD --> DD/MM/YYYY --> 2014-05-21
```

Before the first arrow, you can see the 'original' format. Then the destination format and finally the date we want to convert (Hopefully in the 'original' format).

Output

The program should be able to convert a date from "YYYY-MM-DD" to "DD/MM/YYYY" and vice versa.

If the date format is invalid, or if the values for the day, month, or year are incorrect (e.g., if the day is 32 or the month is 13), the program should return the message "Invalid date".

Example 1

Input

```
YYYY-MM-DD --> DD/MM/YYYY --> 2014-05-21
```

Output

```
21/05/2014
```

Example 2

Input

```
DD/MM/YYYY --> YYYY-MM-DD --> 21/05/2014
```

Output

```
2014-05-21
```

Example 3

Input

```
DD/MM/YYYY --> DD/MM/YYYY --> 20/20/2020
```

Output

```
Invalid date
```

Solutions

Python

```
from datetime import datetime

def convert_date(input_line: str):
    # Split the input by the arrow symbols
    parts = input_line.strip().split("-->")
```



<CODEWARS2026/>
LEÓN

```
if len(parts) != 3:
    return "Invalid date"
source_format = parts[0].strip()
destination_format = parts[1].strip()
date_str = parts[2].strip()
# Map the readable format to Python's datetime format
format_map = {
    "YYYY-MM-DD": "%Y-%m-%d",
    "DD/MM/YYYY": "%d/%m/%Y"
}
# Validate format presence
if source_format not in format_map or destination_format not in
format_map:
    return "Invalid date"
# Try to parse the input date according to the source format
try:
    date_obj = datetime.strptime(date_str,
format_map[source_format])
    # Convert to the destination format
    converted_date =
date_obj.strftime(format_map[destination_format])
    return converted_date
except ValueError:
    return "Invalid date"
# Example usage:
if __name__ == "__main__":
    # You can test with inputs like:
    # "YYYY-MM-DD --> DD/MM/YYYY --> 2024-12-10"
```

<let's code/>





<CODEWARS2026/> LEÓN

```
# "DD/MM/YYYY --> YYYY-MM-DD --> 10/12/2024"  
user_input = input().strip()  
print(convert_date(user_input))
```

C++

```
#include <iostream>  
#include <sstream>  
#include <string>  
#include <vector>  
#include <iomanip>  
#include <ctime>  
using namespace std;  
  
// Función auxiliar para dividir un string por un delimitador  
vector<string> split(const string &str, const string &delimiter) {  
    vector<string> tokens;  
    size_t start = 0, end = 0;  
    while ((end = str.find(delimiter, start)) != string::npos) {  
        tokens.push_back(str.substr(start, end - start));  
        start = end + delimiter.length();  
    }  
    tokens.push_back(str.substr(start));  
    return tokens;  
}  
  
// Verifica si una fecha es válida  
bool isValidDate(int day, int month, int year) {  
    if (year < 1 || month < 1 || month > 12 || day < 1) return false;  
    static const int daysInMonth[] = { 31,28,31,30,31,30,31,31,30,31,30,31  
};  
    int days = daysInMonth[month - 1];
```




<CODEWARS2026/> LEÓN

```
// Año bisiesto
if (month == 2) {
    bool leap = (year % 4 == 0 && (year % 100 != 0 || year % 400 ==
0));
    if (leap) days = 29;
}
return day <= days;
}

string convertDate(const string &inputLine) {
    vector<string> parts = split(inputLine, "-->");
    if (parts.size() != 3) return "Invalid date";
    string sourceFormat = parts[0];
    string destFormat = parts[1];
    string dateStr = parts[2];
    // Eliminar espacios
    auto trim = [](string &s) {
        s.erase(0, s.find_first_not_of(" \t\n\r"));
        s.erase(s.find_last_not_of(" \t\n\r") + 1);
    };
    trim(sourceFormat);
    trim(destFormat);
    trim(dateStr);
    int day, month, year;
    // Leer según el formato de origen
    if (sourceFormat == "YYYY-MM-DD") {
        if (sscanf(dateStr.c_str(), "%d-%d-%d", &year, &month, &day) != 3)
            return "Invalid date";
    } else if (sourceFormat == "DD/MM/YYYY") {
```



<CODEWARS2026/>
LEÓN

```
        if (sscanf(dateStr.c_str(), "%d/%d/%d", &day, &month, &year) != 3)
            return "Invalid date";
    } else {
        return "Invalid date";
    }
    // Validar la fecha
    if (!isValidDate(day, month, year)) return "Invalid date";
    // Convertir según el formato de destino
    ostringstream output;
    if (destFormat == "YYYY-MM-DD") {
        output << setw(4) << setfill('0') << year << "-"
                << setw(2) << setfill('0') << month << "-"
                << setw(2) << setfill('0') << day;
    } else if (destFormat == "DD/MM/YYYY") {
        output << setw(2) << setfill('0') << day << "/"
                << setw(2) << setfill('0') << month << "/"
                << setw(4) << setfill('0') << year;
    } else {
        return "Invalid date";
    }
    return output.str();
}

int main() {
    string inputLine;
    getline(cin, inputLine);
    cout << convertDate(inputLine) << endl;
    return 0;
}
```

Java

```
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.time.format.DateTimeParseException;
import java.util.Scanner;

public class DateConverter {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String input = scanner.nextLine().trim();
        scanner.close();
        System.out.println(convertDate(input));
    }
    public static String convertDate(String inputLine) {
        try {
            String[] parts = inputLine.split("-->");
            if (parts.length != 3) {
                return "Invalid date";
            }
            String sourceFormat = parts[0].trim();
            String destinationFormat = parts[1].trim();
            String dateStr = parts[2].trim();
            DateTimeFormatter sourceFormatter;
            DateTimeFormatter destFormatter;
            switch (sourceFormat) {
                case "YYYY-MM-DD":
                    sourceFormatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");
                    break;
```



<CODEWARS2026/>
LEÓN

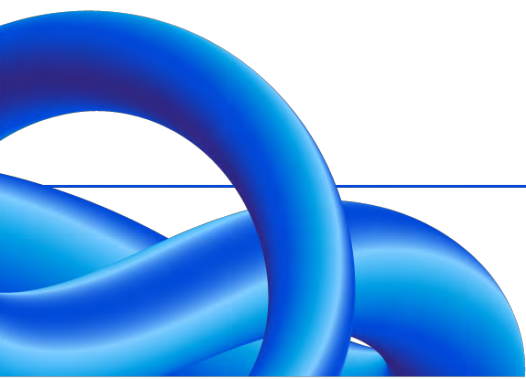
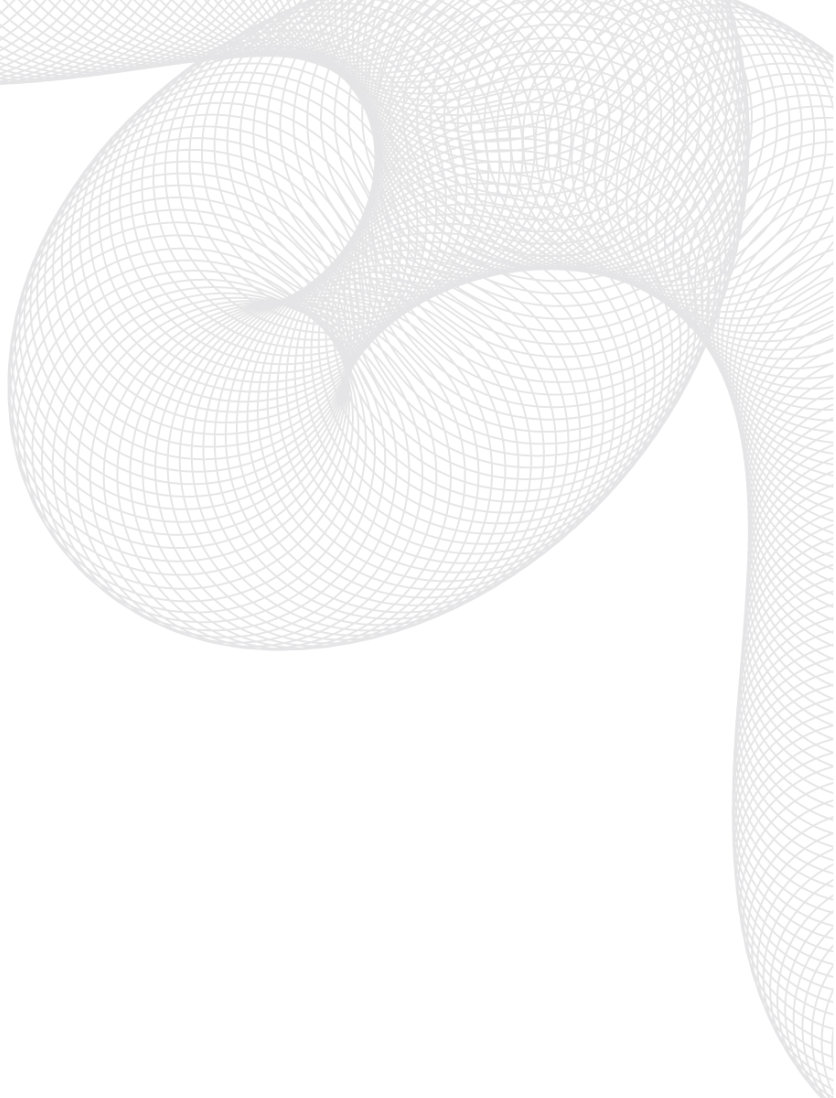
```
        case "DD/MM/YYYY":
            sourceFormatter =
DateTimeFormatter.ofPattern("dd/MM/yyyy");
            break;
        default:
            return "Invalid date";
    }
    switch (destinationFormat) {
        case "YYYY-MM-DD":
            destFormatter = DateTimeFormatter.ofPattern("yyyy-MM-
dd");

            break;
        case "DD/MM/YYYY":
            destFormatter =
DateTimeFormatter.ofPattern("dd/MM/yyyy");
            break;
        default:
            return "Invalid date";
    }
    LocalDate date;
    try {
        date = LocalDate.parse(dateStr, sourceFormatter);
    } catch (DateTimeParseException e) {
        return "Invalid date";
    }
    return date.format(destFormatter);
} catch (Exception e) {
    return "Invalid date";
}
```




<CODEWARS2026/>
LEÓN

```
}  
}  
}
```



<let's code/>



13

Yo soy tu primo

8 points

Introduction

In the Cloud City, Luke Skywalker faces Darth Vader in a decisive duel. In the midst of the battle, Darth Vader is about to reveal a shocking secret: they could be cousins!

But it's not a matter of blood relation; it's a mathematical one. Your mission is to write a program that helps Luke discover whether Darth Vader is telling the truth.



Exercise

The program should do the following:

- > Receive a number that represents the Force power of Luke or Darth Vader.
- > Determine if the entered number is prime.

-> If the number is prime, display the message: "It's true! This number is prime. The Force connection is real."

-> If the number is not prime, display: "The number is not prime. Darth Vader is lying."

-> If the number is 0, the program should finish with the message: "May the Force be with you, young padawan."

-> Otherwise, write "The entered number is incorrect, young padawan."

Input

The program will read a single text line.

Output

The output will be a single text line following the instructions given earlier.

Example 1

Input

7

Output

It's true! The number 7 is prime. The connection to the Force is real.

Example 2

Input

10

Output

The number 10 is not prime. Darth Vader lies.

Example 3

Input

0

Output

May the Force be with you, young padawan.

Example 4

Input

abc

Output

The entered number is incorrect, young padawan.

Solutions

Python

```
def is_prime(number):  
    if number <= 1:  
        return False  
    for i in range(2, int(number ** 0.5) + 1):  
        if number % i == 0:  
            return False  
    return True  
  
try:  
    number = int(input())  
    if number == 0:
```



```
        print("May the Force be with you, young padawan.")
    elif is_prime(number):
        print(f"It's true! The number {number} is prime. The
connection to the Force is real.")
    else:
        print(f"The number {number} is not prime. Darth Vader lies.")
except ValueError:
    print("The entered number is incorrect, young padawan.")
```

C++

```
#include <iostream>
#include <cmath>
using namespace std;
bool isPrime(long long number) {
    if (number <= 1) return false;
    for (long long i = 2; i <= sqrt(number); ++i) {
        if (number % i == 0) return false;
    }
    return true;
}
int main() {
    long long number;
    if (!(cin >> number)) {
        cout << "The entered number is incorrect, young padawan." <<
endl;

        cin.clear();
        cin.ignore(10000, '\n');
    }
    else if (number == 0) {
```



<CODEWARS2026/>
LEÓN

```
        cout << "May the Force be with you, young padawan." << endl;
    }
    else if (isPrime(number)) {
        cout << "It's true! The number " << number << " is prime. The
connection to the Force is real." << endl;
    } else {
        cout << "The number " << number << " is not prime. Darth Vader
lies." << endl;
    }
    return 0;
}
```

Java

```
import java.util.InputMismatchException;
import java.util.Scanner;
public class Yo_Soy_Tu_Primo
{
    /**
     * Check if a number is prime
     * @param number number to check
     * @return boolean if the number is prime or not
     */
    public static boolean isPrime(long number) {
        if (number <= 1) {
            return false;
        }
        for (int i = 2; i <= Math.sqrt(number); i++) {
            if (number % i == 0) {
                return false;
            }
        }
    }
}
```



<CODEWARS2026/>
LEÓN

```
        }
    }
    return true;
}

public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    try{
        long number;
        number = scanner.nextLong();
        if (number == 0) {
            System.out.println("May the Force be with you, young
padawan.");
        }
        else if (isPrime(number)) {
            System.out.println("It's true! The number " + number +
" is prime. The connection to the Force is real.");
        } else {
            System.out.println("The number " + number + " is not
prime. Darth Vader lies.");
        }
    } catch (InputMismatchException e) {
        System.out.println("The entered number is incorrect, young
padawan.");
        scanner.nextLine();
    }
}
}
```


14

Formula 1 Tactical Manager

9 points

Introduction

You are an F1 expert strategy manager that needs to calculate the optimal tire strategy for a Formula 1 race. In Formula 1, there are three types of tires available, each with different characteristics:

- Soft tires: Faster lap times but wear out quickly
- Medium tires: Balanced performance and durability
- Hard tires: Slower lap times but last longer

Each tire type has two key parameters:

1. Lap time (in seconds): The time it takes to complete one lap with these tires
2. Maximum laps: The maximum number of laps these tires can last before they must be changed. There is no races with more than 30 laps.



Race Rules

1. Mandatory pit stop: At least one pit stop must be made during the race

2. Pit stop time: Each pit stop takes a fixed time of 20 seconds
3. Tire variety rule: At least TWO different tire types must be used during the race
4. Tire reuse: The same tire type can be used multiple times (in different stints)

Exercise

Write a program to find the tire strategy that minimizes the total race time (lap times + pit stop penalties).

Input

The program receives input from the keyboard (stdin) in the following format:

```
<soft_laptime>,<soft_maxlaps>-<medium_laptime>,<medium_maxlaps>-  
<hard_laptime>,<hard_maxlaps>  
<total_laps>
```

Where:

- Tire parameters are separated by commas , within each type
- Different tire types are separated by hyphens -
- The format is: *soft-medium-hard*
- Second line contains the total number of laps in the race

For instance:

```
120.13,10-130.26,20-140.53,30  
30
```

This means:

- Soft tires: 120.13 seconds per lap, maximum 10 laps
- Medium tires: 130.26 seconds per lap, maximum 20 laps
- Hard tires: 140.53 seconds per lap, maximum 30 laps

- Race distance: 30 laps

Output

The program should output to the screen (stdout):

```
<tire1>(<laps1>),<tire2>(<laps2>), ... <tireN>(<lapsN>)
```

Where:

- <tire> is one of: S (for Soft), M (for Medium), or H (For Hard)
- <laps> is the number of laps on that tire compound

With the previous input, the output would be:

```
S(9),M(1),S(10),S(10)
```

Example 1

Input

```
120.13,10-130.26,20-140.53,30  
30
```

Output

```
S(9),M(1),S(10),S(10)
```

Example 2

Input

```
90.5,5-95.2,10-100.8,15  
12
```

Output

```
S(5),S(5),M(2)
```

Solutions

Python

```
import sys
from typing import List, Tuple
class Tire:
    def __init__(self, name: str, lap_time: float = 0.0, max_laps: int =
0):
        self.name = name
        self.lap_time = lap_time
        self.max_laps = max_laps
class Strategy:
    def __init__(self):
        self.stints: List[Tuple[int, int]] = [] # List of
(tire_type_index, number_of_laps)
        self.total_time: float = float('inf')
PIT_STOP_TIME = 20.0
def parse_input(input_str: str) -> List[Tire]:
    """Parse input string to extract tire parameters"""
    tires = [
        Tire("S"),
        Tire("M"),
        Tire("H")
    ]
    tokens = input_str.split('-')
    for idx, token in enumerate(tokens[:3]):
        parts = token.split(',')
        tires[idx].lap_time = float(parts[0])
```



<CODEWARS2026/>
LEÓN

```
        tires[idx].max_laps = int(parts[1])
    return tires

def uses_at_least_two_types(stints: List[Tuple[int, int]]) -> bool:
    """Check if a strategy uses at least two different tire types"""
    if not stints:
        return False
    first_type = stints[0][0]
    for stint in stints:
        if stint[0] != first_type:
            return True
    return False

def calculate_time(tires: List[Tire], stints: List[Tuple[int, int]]) ->
float:
    """Calculate total time for a given strategy"""
    total_time = 0.0
    for tire_type, laps in stints:
        total_time += tires[tire_type].lap_time * laps
    total_time += (len(stints) - 1) * PIT_STOP_TIME
    return total_time

def generate_strategies(tires: List[Tire], total_laps: int, current_lap:
int,
                        current_stints: List[Tuple[int, int]],
best_strategy: Strategy) -> None:
    """Recursive function to generate all valid strategies"""
    # Base case: we've covered all laps
    if current_lap == total_laps:
        # Check if strategy is valid (at least 2 tire types and at least 1
        pit stop)
```



```
        if len(current_stints) >= 2 and
uses_at_least_two_types(current_stints):
            time = calculate_time(tires, current_stints)
            if time < best_strategy.total_time:
                best_strategy.stints = current_stints.copy()
                best_strategy.total_time = time
    return

# Try each tire type
for tire_type in range(3):
    # Try different stint lengths (1 to min(maxLaps, remaining laps))
    remaining_laps = total_laps - current_lap
    max_stint_laps = min(tires[tire_type].max_laps, remaining_laps)
    for stint_laps in range(1, max_stint_laps + 1):
        # Pruning: if we've already used too much time, skip
        current_stints.append((tire_type, stint_laps))
        # Only recurse if this could potentially be better
        current_time = calculate_time(tires, current_stints)
        remaining_after_stint = total_laps - current_lap - stint_laps
        # Simple lower bound: remaining laps with fastest tire
        min_possible_time = current_time
        if remaining_after_stint > 0:
            fastest_lap_time = min(tire.lap_time for tire in tires)
            min_possible_time += remaining_after_stint *
fastest_lap_time + PIT_STOP_TIME
        if min_possible_time < best_strategy.total_time:
            generate_strategies(tires, total_laps, current_lap +
stint_laps,

                                current_stints, best_strategy)
```



<CODEWARS2026/> LEÓN

```
        current_stints.pop()

def main():
    # Read input
    tire_input = input().strip()
    total_laps = int(input().strip())
    # Parse tire data
    tires = parse_input(tire_input)
    # Find best strategy
    best_strategy = Strategy()
    current_stints: List[Tuple[int, int]] = []
    generate_strategies(tires, total_laps, 0, current_stints,
best_strategy)
    # Output result
    result = []
    for tire_type, laps in best_strategy.stints:
        result.append(f"{tires[tire_type].name}({laps})")
    print(",".join(result))
if __name__ == "__main__":
    main()
```

C++

```
#include <iostream>
#include <string>
#include <vector>
#include <sstream>
#include <limits>
#include <algorithm>
using namespace std;
struct Tire {
```



<CODEWARS2026/> LEÓN

```
    string name;
    double lapTime;
    int maxLaps;
};
struct Strategy {
    vector<pair<int, int>> stints; // pair of (tire_type_index,
number_of_laps)
    double totalTime;
    Strategy() : totalTime(numeric_limits<double>::max()) {}
};
const double PIT_STOP_TIME = 20.0;
// Parse input string to extract tire parameters
vector<Tire> parseInput(const string& input) {
    vector<Tire> tires(3);
    tires[0].name = "S";
    tires[1].name = "M";
    tires[2].name = "H";
    stringstream ss(input);
    string token;
    int idx = 0;
    while (getline(ss, token, '-') && idx < 3) {
        size_t commaPos = token.find(',');
        tires[idx].lapTime = stod(token.substr(0, commaPos));
        tires[idx].maxLaps = stoi(token.substr(commaPos + 1));
        idx++;
    }
    return tires;
}
```



```
// Check if a strategy uses at least two different tire types
bool usesAtLeastTwoTypes(const vector<pair<int, int>>& stints) {
    if (stints.empty()) return false;
    int firstType = stints[0].first;
    for (const auto& stint : stints) {
        if (stint.first != firstType) {
            return true;
        }
    }
    return false;
}

// Calculate total time for a given strategy
double calculateTime(const vector<Tire>& tires, const vector<pair<int,
int>>& stints) {
    double totalTime = 0.0;
    for (const auto& stint : stints) {
        int tireType = stint.first;
        int laps = stint.second;
        totalTime += tires[tireType].lapTime * laps;
    }
    totalTime += (stints.size() - 1) * PIT_STOP_TIME;
    return totalTime;
}

// Recursive function to generate all valid strategies
void generateStrategies(const vector<Tire>& tires, int totalLaps, int
currentLap,
                        vector<pair<int, int>>& currentStints, Strategy&
bestStrategy) {
```




```
// Base case: we've covered all laps
if (currentLap == totalLaps) {
    // Check if strategy is valid (at least 2 tire types and at least
1 pit stop)
    if (currentStints.size() >= 2 &&
usesAtLeastTwoTypes(currentStints)) {
        double time = calculateTime(tires, currentStints);
        if (time < bestStrategy.totalTime) {
            bestStrategy.stints = currentStints;
            bestStrategy.totalTime = time;
        }
    }
    return;
}

// Try each tire type
for (int tireType = 0; tireType < 3; tireType++) {
    // Try different stint lengths (1 to min(maxLaps, remaining laps))
    int remainingLaps = totalLaps - currentLap;
    int maxStintLaps = min(tires[tireType].maxLaps, remainingLaps);
    for (int stintLaps = 1; stintLaps <= maxStintLaps; stintLaps++) {
        // Pruning: if we've already used too much time, skip
        currentStints.push_back({tireType, stintLaps});
        // Only recurse if this could potentially be better
        double currentTime = calculateTime(tires, currentStints);
        int remainingAfterStint = totalLaps - currentLap - stintLaps;
        // Simple lower bound: remaining laps with fastest tire
        double minPossibleTime = currentTime;
        if (remainingAfterStint > 0) {
```



<CODEWARS2026/>
LEÓN

```
        double fastestLapTime = min({tires[0].lapTime,
tires[1].lapTime, tires[2].lapTime});
        minPossibleTime += remainingAfterStint * fastestLapTime +
PIT_STOP_TIME;
    }
    if (minPossibleTime < bestStrategy.totalTime) {
        generateStrategies(tires, totallaps, currentLap +
stintLaps, currentStints, bestStrategy);
    }
    currentStints.pop_back();
}
}

int main() {
    string tireInput;
    int totallaps;
    // Read input
    getline(cin, tireInput);
    cin >> totallaps;
    // Parse tire data
    vector<Tire> tires = parseInput(tireInput);
    // Find best strategy
    Strategy bestStrategy;
    vector<pair<int, int>> currentStints;
    generateStrategies(tires, totallaps, 0, currentStints, bestStrategy);
    // Output result
    for (size_t i = 0; i < bestStrategy.stints.size(); i++) {
        int tireType = bestStrategy.stints[i].first;
```



<CODEWARS2026/>
LEÓN

```
int laps = bestStrategy.stints[i].second;
cout << tires[tireType].name << "(" << laps << ")";
if (i < bestStrategy.stints.size() - 1) {
    cout << ",";
}
}
cout << endl;
return 0;
}
```

Java

```
import java.util.*;
public class F1Strategy {
    static class Tire {
        String name;
        double lapTime;
        int maxLaps;
        Tire(String name, double lapTime, int maxLaps) {
            this.name = name;
            this.lapTime = lapTime;
            this.maxLaps = maxLaps;
        }
    }
    static class Stint {
        int tireTypeIndex;
        int numberOfLaps;
        Stint(int tireTypeIndex, int numberOfLaps) {
            this.tireTypeIndex = tireTypeIndex;
            this.numberOfLaps = numberOfLaps;
        }
    }
}
```



<CODEWARS2026/>
LEÓN

```
    }  
}  
static class Strategy {  
    List<Stint> stints;  
    double totalTime;  
    Strategy() {  
        this.stints = new ArrayList<>();  
        this.totalTime = Double.MAX_VALUE;  
    }  
}  
  
static final double PIT_STOP_TIME = 20.0;  
// Parse input string to extract tire parameters  
static List<Tire> parseInput(String input) {  
    List<Tire> tires = new ArrayList<>();  
    tires.add(new Tire("S", 0.0, 0));  
    tires.add(new Tire("M", 0.0, 0));  
    tires.add(new Tire("H", 0.0, 0));  
    String[] tokens = input.split("-");  
    for (int idx = 0; idx < Math.min(tokens.length, 3); idx++) {  
        String[] parts = tokens[idx].split(",");  
        tires.get(idx).lapTime = Double.parseDouble(parts[0]);  
        tires.get(idx).maxLaps = Integer.parseInt(parts[1]);  
    }  
    return tires;  
}  
  
// Check if a strategy uses at least two different tire types  
static boolean usesAtLeastTwoTypes(List<Stint> stints) {  
    if (stints.isEmpty()) return false;
```




<CODEWARS2026/>
LEÓN

```
int firstType = stints.get(0).tireTypeIndex;
for (Stint stint : stints) {
    if (stint.tireTypeIndex != firstType) {
        return true;
    }
}
return false;
}

// Calculate total time for a given strategy
static double calculateTime(List<Tire> tires, List<Stint> stints) {
    double totalTime = 0.0;
    for (Stint stint : stints) {
        int tireType = stint.tireTypeIndex;
        int laps = stint.numberOfLaps;
        totalTime += tires.get(tireType).lapTime * laps;
    }
    totalTime += (stints.size() - 1) * PIT_STOP_TIME;
    return totalTime;
}

// Recursive function to generate all valid strategies
static void generateStrategies(List<Tire> tires, int totalLaps, int
currentLap,

                                List<Stint> currentStints, Strategy
bestStrategy) {
    // Base case: we've covered all laps
    if (currentLap == totalLaps) {
        // Check if strategy is valid (at least 2 tire types and at
least 1 pit stop)
```

```
        if (currentStints.size() >= 2 &&
usesAtLeastTwoTypes(currentStints)) {
            double time = calculateTime(tires, currentStints);
            if (time < bestStrategy.totalTime) {
                bestStrategy.stints = new ArrayList<>(currentStints);
                bestStrategy.totalTime = time;
            }
        }
        return;
    }
    // Try each tire type
    for (int tireType = 0; tireType < 3; tireType++) {
        // Try different stint lengths (1 to min(maxLaps, remaining
laps))

        int remainingLaps = totallaps - currentLap;
        int maxStintLaps = Math.min(tires.get(tireType).maxLaps,
remainingLaps);
        for (int stintLaps = 1; stintLaps <= maxStintLaps;
stintLaps++) {
            // Pruning: if we've already used too much time, skip
currentStints.add(new Stint(tireType, stintLaps));
            // Only recurse if this could potentially be better
            double currentTime = calculateTime(tires, currentStints);
            int remainingAfterStint = totallaps - currentLap -
stintLaps;

            // Simple lower bound: remaining laps with fastest tire
            double minPossibleTime = currentTime;
            if (remainingAfterStint > 0) {
```

```
        double fastestLapTime =  
Math.min(Math.min(tires.get(0).lapTime,  
  
tires.get(1).lapTime),  
  
tires.get(2).lapTime);  
        minPossibleTime += remainingAfterStint *  
fastestLapTime + PIT_STOP_TIME;  
    }  
    if (minPossibleTime < bestStrategy.totalTime) {  
        generateStrategies(tires, totalLaps, currentLap +  
stintLaps,  
  
currentStints, bestStrategy);  
    }  
    currentStints.remove(currentStints.size() - 1);  
}  
}  
}  
  
public static void main(String[] args) {  
    Scanner scanner = new Scanner(System.in);  
    // Read input  
    String tireInput = scanner.nextLine();  
    int totalLaps = scanner.nextInt();  
    // Parse tire data  
    List<Tire> tires = parseInput(tireInput);  
    // Find best strategy  
    Strategy bestStrategy = new Strategy();  
    List<Stint> currentStints = new ArrayList<>();
```



<CODEWARS2026/>
LEÓN

```
        generateStrategies(tires, totallaps, 0, currentStints,
bestStrategy);
    // Output result
    for (int i = 0; i < bestStrategy.stints.size(); i++) {
        Stint stint = bestStrategy.stints.get(i);
        int tireType = stint.tireTypeIndex;
        int laps = stint.numberOfLaps;
        System.out.print(tires.get(tireType).name + "(" + laps + ")");
        if (i < bestStrategy.stints.size() - 1) {
            System.out.print(",");
        }
    }
    System.out.println();
    scanner.close();
}
}
```

<let's code/>



15

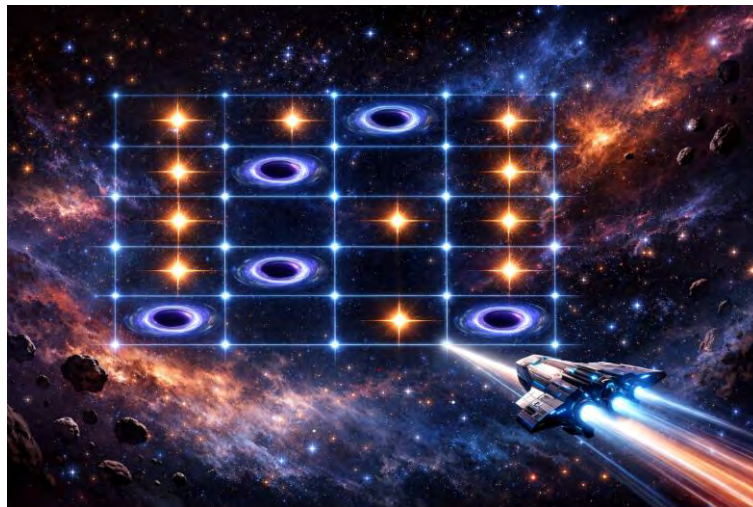
Interstellar Travel Organization

9 points

Introduction

In a distant galaxy, an advanced civilization has developed the ability to travel at the speed of light. However, their path through space is filled with stars (marked as '*') and wormholes (marked as 'O'). The distance between each star is measured in parsecs, indicated by the number preceding each star (e.g., 2* means a distance of 2 parsecs).

Each wormhole (O) halves the total travel time accumulated up to that point. The goal is to traverse the entire grid, either row by row or column by column, while minimizing travel time.



Exercise

Your task is to develop a program that:

1. Calculate the total travel time for each row and each column.
2. Determine the optimal path that minimizes the time required to travel from one edge of the grid to the opposite side.

Consider that traveling at the speed of light takes 1 year per parsec unless reduced by a wormhole. Can you find the fastest route through this cosmic maze?

Input

The input will be a galaxy represented by an array of 5x5 stars or wormholes like the following:

```
2* 2* 0 3* 2*
1* 3* 0 1* 3*
0 0 0 0 0
2* 3* 0 4* 6*
4* 5* 0 4* 5*
```

As indicated the array is filled with stars (marked as *) and wormholes (marked as O). The distance between each star is measured in parsecs, indicated by the number preceding each star (e.g., 2* means a distance of 2 parsecs).

The galaxy can only be traversed following the stars and wormholes in a row or in a column (unless it's filled by wormholes only).

The calculations in this case will be, if we follow the rows of the the array the required travel time will be:

Row	Calculation	Result
1	7.0 --> Because it's 2 + 2 , now we divide by 2 --> result is 2 + 3 + 2 = 7	7.0
2	6.0 --> Because it's 1 + 3 , now we divide by 2 --> result is 2 + 1 + 3 = 6	6.0
3	0.0 --> 0 + 0 + 0 + 0 + 0 = 0	0.0
4	12.5 --> 2 + 3 , now divide by 2 --> result is 2.5 + 4 + 6 = 12.5	12.5
5	13.5 --> ...	13.5

And for the columns:

Column	Result
1	7.5
2	10.5

```
3 0.0
4 10.0
5 13.5
```

Output

Therefore the output will be:

The best possible route would be row 2, with an interstellar space flight time of 6.0 unit

Note: If any of the values is fractional, only one decimal will be shown. This value should be rounded (4.88 would 4.9 and 4.82 would be 4.8). If the second decimal of flight time is 5 (for instance 4.25), note that the expected output would be 4.2.

Example 1

Input

```
2* 2* 0 3* 2*
1* 3* 0 1* 3*
0 0 0 0 0
2* 3* 0 4* 6*
4* 5* 0 4* 5*
```

Output

The best possible route would be row 2, with an interstellar space flight time of 6.0 unit

Example 2

Input

```
2* 2* 0 3* 0
1* 0 3* 0 3*
0 0 9* 0 1*
2* 3* 0 4* 6*
4* 5* 0 4* 5*
```

Output

The best possible route would be row 1, with an interstellar space flight time of 2.5 unit

Solutions

Python

```
def calculate_travel_time(line):
    total_time = 0
    all_wormholes = True
    for item in line:
        if item == 'O': # Wormhole found
            total_time /= 2 # Halve the current total time
        else:
            # Extract the number from the format 'X*' where X is a number
            parsec_distance = int(item[:-1]) # Remove '*' and convert to
integer
            total_time += parsec_distance
            all_wormholes = False # Found a star, so it's not all
wormholes
        # If the line was entirely wormholes, consider the time as invalid
(infinity)
        if all_wormholes:
            return float('inf') # Return a large value to indicate an invalid
path
    return total_time
def solve_galaxy(grid):
    # 5x5 grid
    rows = len(grid)
    cols = len(grid[0])
```




<CODEWARS2026/>
LEÓN

```
min_time = float('inf')
min_path = None
# Calculate the travel times for each row
for i in range(rows):
    row_time = calculate_travel_time(grid[i])
    if row_time < min_time:
        min_time = row_time
        min_path = f"row {i+1}"
# Calculate the travel times for each column
for j in range(cols):
    column = [grid[i][j] for i in range(rows)]
    col_time = calculate_travel_time(column)
    if col_time < min_time:
        min_time = col_time
        min_path = f"column {j+1}"
# If the best time is infinity, it means we had an invalid path with
only wormholes.
if min_time == float('inf'):
    return "No valid path exists", 0
return min_path, min_time
# Read the galaxy grid from standard input
def read_input():
    galaxy = []
    for i in range(5):
        line = input() # Read the line directly
        galaxy.append(line.split()) # Split by spaces and append to the
galaxy list
    return galaxy
```



<CODEWARS2026/> LEÓN

```
# Main execution
galaxy = read_input() # Input is read here, no extra print statements
# Solve the galaxy problem
best_path, best_time = solve_galaxy(galaxy)
# Print the result
if best_time == 0:
    print("No valid path exists")
else:
    print(f"The best possible route would be {best_path}, with an
interstellar space flight time of {best_time:.1f} unit")
```

C++

```
#include <iostream>
#include <vector>
#include <string>
#include <sstream>
#include <limits>
#include <iomanip>
#include <cmath>
#include <cfenv>
using namespace std;
// Function to calculate the travel time for a row or column
double calculate_travel_time(const vector<string>& line) {
    double total_time = 0;
    bool all_wormholes = true;
    for (const string& item : line) {
        if (item == "0") {
            total_time /= 2; // Halve the current total time
        } else {
```



```
        int parsec_distance = stoi(item.substr(0, item.length() - 1));
// Remove '*' and convert to integer
        total_time += parsec_distance;
        all_wormholes = false; // Found a star, so it's not all
wormholes
    }
}
// If the line was entirely wormholes, consider the time as invalid
(infinity)
    if (all_wormholes) {
        return numeric_limits<double>::infinity(); // Return a large
value to indicate an invalid path
    }
    return total_time;
}
// Function to solve the galaxy problem
pair<string, double> solve_galaxy(const vector<vector<string>>& grid) {
    int rows = grid.size();
    int cols = grid[0].size();
    double min_time = numeric_limits<double>::infinity();
    string min_path;
    // Calculate the travel times for each row
    for (int i = 0; i < rows; ++i) {
        double row_time = calculate_travel_time(grid[i]);
        if (row_time < min_time) {
            min_time = row_time;
            min_path = "row " + to_string(i + 1);
        }
    }
```



<CODEWARS2026/>
LEÓN

```
}  
// Calculate the travel times for each column  
for (int j = 0; j < cols; ++j) {  
    vector<string> column;  
    for (int i = 0; i < rows; ++i) {  
        column.push_back(grid[i][j]);  
    }  
    double col_time = calculate_travel_time(column);  
    if (col_time < min_time) {  
        min_time = col_time;  
        min_path = "column " + to_string(j + 1);  
    }  
}  
  
// If the best time is infinity, it means we had an invalid path with  
only wormholes.  
if (min_time == numeric_limits<double>::infinity()) {  
    return {"No valid path exists", 0};  
}  
return {min_path, min_time};  
}  
  
// Function to read the input  
vector<vector<string>> read_input() {  
    vector<vector<string>> galaxy(5);  
    string line;  
    for (int i = 0; i < 5; ++i) {  
        getline(cin, line);  
        stringstream ss(line);  
        string item;
```




<CODEWARS2026/> LEÓN

```
        while (ss >> item) {
            galaxy[i].push_back(item);
        }
    }
    return galaxy;
}

// Main function
int main() {
    vector<vector<string>> galaxy = read_input(); // Read the input grid
    // Solve the galaxy problem
    auto [best_path, best_time] = solve_galaxy(galaxy);
    std::fesetround(FE_TONEAREST); // redondeo al par más cercano
    (ties-to-even)
    double print_time = std::nearbyint(best_time * 10.0) / 10.0;
    // Print the result
    if (best_time == 0) {
        cout << "No valid path exists" << endl;
    } else {
        cout << "The best possible route would be " << best_path << ",
with an interstellar space flight time of " << std::fixed <<
std::setprecision(1) << print_time << " unit" << endl;
    }
    return 0;
}
```

Java

```
import java.util.*;
import java.math.BigDecimal;
import java.math.RoundingMode;
```

```
public class GalaxyTravel {  
    // Function to calculate the travel time for a row or column  
    public static double calculateTravelTime(String[] line) {  
        double totalTime = 0;  
        boolean allWormholes = true;  
        for (String item : line) {  
            if (item.trim().equals("0")) {  
                totalTime /= 2; // Halve the current total time  
            } else if (item.trim().matches("[1-9]+\\*$")) {  
                int parsecDistance = Integer.parseInt(item.substring(0,  
item.length() - 1)); // Remove '*' and convert to integer  
                totalTime += parsecDistance;  
                allWormholes = false; // Found a star, so it's not all  
wormholes  
            }  
            else  
            {  
                // If the line has unknown elements, consider the time as  
invalid (infinity)  
                return Double.POSITIVE_INFINITY;  
            }  
        }  
        // If the line was entirely wormholes, consider the time as  
invalid (infinity)  
        if (allWormholes) {  
            return Double.POSITIVE_INFINITY; // Return a large value to  
indicate an invalid path  
        }  
    }  
}
```



<CODEWARS2026/>
LEÓN

```
        return totalTime;
    }
    // Function to solve the galaxy problem
    public static String[] solveGalaxy(String[][] grid) {
        int rows = grid.length;
        int cols = grid[0].length;
        double minTime = Double.POSITIVE_INFINITY;
        String minPath = "";
        // Calculate the travel times for each row
        for (int i = 0; i < rows; i++) {
            double rowTime = calculateTravelTime(grid[i]);
            if (rowTime < minTime) {
                minTime = rowTime;
                minPath = "row " + (i + 1);
            }
        }
        // Calculate the travel times for each column
        for (int j = 0; j < cols; j++) {
            String[] column = new String[rows];
            for (int i = 0; i < rows; i++) {
                column[i] = grid[i][j];
            }
            double colTime = calculateTravelTime(column);
            if (colTime < minTime) {
                minTime = colTime;
                minPath = "column " + (j + 1);
            }
        }
    }
```

```
// If the best time is infinity, it means we had an invalid path
with only wormholes.
    if (minTime == Double.POSITIVE_INFINITY) {
        return new String[]{"No valid path exists", "0"};
    }
    BigDecimal bd = BigDecimal.valueOf(minTime).setScale(1,
RoundingMode.HALF_EVEN);
    return new String[]{minPath, bd.toPlainString()};
    //return new String[]{minPath, String.format(Locale.ENGLISH,
"%1f", minTime)};
}
// Function to read the input
public static String[][] readInput() {
    Scanner sc = new Scanner(System.in);
    String[][] galaxy = new String[5][5];
    for (int i = 0; i < 5; i++) {
        String line = sc.nextLine();
        // Split by spaces and store in the galaxy array avoiding add
empty elements
        galaxy[i] = line.split("[ ]+", 0);
    }
    return galaxy;
}
public static void main(String[] args) {
    String[][] galaxy = readInput(); // Read the input grid
    // Solve the galaxy problem
    String[] result = solveGalaxy(galaxy);
    // Print the result
```




<CODEWARS2026/>
LEÓN

```
        if (result[0].equals("No valid path exists")) {  
            System.out.println(result[0]);  
        } else {  
            System.out.println("The best possible route would be " +  
result[0] +  
                                ", with an interstellar space flight time  
of " + result[1] + " unit");  
        }  
    }  
}
```

<let's code/>



16

La cartografía del vacío

10 points

Introduction

Los Ixianos, una antigua civilización galáctica dedicada a la minería, explotación, transformación y comercio de recursos obtenidos en los anillos de asteroides, estaban maravillados con su nuevo descubrimiento: una sonda capaz de atravesar el campo de escombros que rodea a un sistema estelar sin sufrir daño alguno.

Para ellos, este hallazgo no era simplemente un avance tecnológico: representaba una posibilidad legal y cultural.

En la sociedad ixiana, interferir con mundos que albergan vida o con civilizaciones en estado primitivo es considerado un crimen contra la existencia. Por ello, hace milenios se estableció una **ley universal**:

Solo es legal extraer recursos de los cinturones de asteroides circundantes a los sistemas estelares.

De este modo, los Ixianos podían seguir prosperando sin infringir las normas que protegen el desarrollo de otras especies.

La sonda que habían encontrado tenía un nombre familiar para nosotros: **Voyager**.

Pero para los Ixianos, que abandonaron hace milenios el lenguaje hablado y escrito debido a su evolución hacia una civilización telepática, aquella estructura era algo completamente diferente: un artefacto incomprensible, lleno de símbolos, marcas y trazos cuyo propósito ignoraban.



Lo que para la humanidad era un mensaje al cosmos, para los Ixianos era un enigma visual.

Lo único que lograban “percibir” eran los vacíos, huecos y silencios entre las marcas inscritas en el artefacto.

Exercise

Como investigador humano que colabora con una delegación ixiana, debes desarrollar un Analizador Visual de Letras que permita traducir un texto humano al sistema de clasificación perceptiva de esta especie.

El programa deberá:

1. Recibir un texto (una o varias líneas).
2. Analizar cada carácter visible y clasificarlo según el tipo de vacío visual que genera:
 - o Encerrado → una cavidad completamente cerrada (o, a, b, d, e, g...) → Para los Ixianos: “camara sagrada de vacío”

- o Semiabierto → cavidad parcialmente abierta (c, m, n, r, s, w...) → Para los Ixianos: “puerta de vacío”
 - o Abierto → sin cavidad interna (i, l, t, v, x...) → Para los Ixianos: “flujo libre del cosmos”
3. Registrar línea, columna y posición absoluta de cada carácter.
 4. Generar un informe alienígena que les permita comprender la estructura del mensaje.
 5. Acompañar el programa con tests unitarios, pues los Ixianos no aceptarán un análisis defectuoso.

El listado completo de categorías de letras ixianas es el siguiente:

- **encerrado** : símbolos alfanuméricos que tienen un hueco interior cerrado
Ej.: a, b, d, e, g, o, p, q, A, B, D, O, P, Q, R, Ø, 4, 6, 8, 9
- **semiabierto_** : símbolos alfanuméricos cuya forma sugiere uno o varios huecos parcialmente abiertos.

Ej.: c, f, h, k, m, n, r, s, u, y, w, C, F, H, K, M, N, R, S, U, Y, W, E, G, 2, 3, 5)

- **abierto** : símbolos alfanuméricos que no forman hueco interior significativo.

Ej.: i, j, l, t, v, x, z, I, J, L, T, V, 16, Z, 1, 7

- **ninguno** : espacios, saltos de línea, signos de puntuación, etc.

Input

Al ejecutar el programa, el programa leera de la entrada estandar hasta encontrar una linea en blanco.

Output

Al ejecutar el programa deberá producirse un informe que represente el texto humano tal como lo percibe la mente ixiana:

no como una secuencia de fonemas, sino como un mapa geométrico de cavidades.

El objetivo final es que la delegación pueda, por primera vez en milenios, "leer" un mensaje en un lenguaje distinto al suyo.

La estructura del informe será la siguiente:

- Una primera sección llamada "Resumen Visual" con el número total de letras de cada tipo de hueco: "encerrado", "semiabierto" y "abierto". Algo así:

```
--- Resumen visual ---  
Letras de hueco ENCERRADO : 5  
Letras de hueco SEMIABIERTO : 2  
Letras de hueco ABIERTO : 2
```

- Una segunda sección con el listado de letras indicando: La línea y la columna donde aparecen dentro del texto, la letra y su clasificación y el significado para los ixianos.

Sería algo así:

```
--- Detalle de letras con vacio significativo ---  
Linea 1, Columna 1 | 'h' -> SEMIABIERTO | puerta de vacio  
Linea 1, Columna 2 | 'o' -> ENCERRADO | camara sagrada de vacio  
Linea 1, Columna 3 | 'l' -> ABIERTO | flujo libre del cosmos  
Linea 1, Columna 4 | 'a' -> ENCERRADO | camara sagrada de vacio  
Linea 2, Columna 1 | 'a' -> ENCERRADO | camara sagrada de vacio  
Linea 2, Columna 2 | 'm' -> SEMIABIERTO | puerta de vacio  
Linea 2, Columna 3 | 'i' -> ABIERTO | flujo libre del cosmos  
Linea 2, Columna 4 | 'g' -> ENCERRADO | camara sagrada de vacio  
Linea 2, Columna 5 | 'o' -> ENCERRADO | camara sagrada de vacio
```

Example 1

Input

```
hola  
amigo
```

Output

```
--- Resumen visual ---
Letras de hueco ENCERRADO   : 5
Letras de hueco SEMIABIERTO : 2
Letras de hueco ABIERTO    : 2
--- Detalle de letras con vacio significativo ---
Linea  1, Columna  1 | 'h' -> SEMIABIERTO | puerta de vacio
Linea  1, Columna  2 | 'o' -> ENCERRADO   | camara sagrada de vacio
Linea  1, Columna  3 | 'l' -> ABIERTO    | flujo libre del cosmos
Linea  1, Columna  4 | 'a' -> ENCERRADO   | camara sagrada de vacio
Linea  2, Columna  1 | 'a' -> ENCERRADO   | camara sagrada de vacio
Linea  2, Columna  2 | 'm' -> SEMIABIERTO | puerta de vacio
Linea  2, Columna  3 | 'i' -> ABIERTO    | flujo libre del cosmos
Linea  2, Columna  4 | 'g' -> ENCERRADO   | camara sagrada de vacio
Linea  2, Columna  5 | 'o' -> ENCERRADO   | camara sagrada de vacio
```

Example 2

Input

```
Bienvenido
a HP Codewars
2026
```

Output

```
--- Resumen visual ---
Letras de hueco ENCERRADO   : 13
Letras de hueco SEMIABIERTO : 9
Letras de hueco ABIERTO    : 3
--- Detalle de letras con vacio significativo ---
Linea  1, Columna  1 | 'B' -> ENCERRADO   | camara sagrada de vacio
Linea  1, Columna  2 | 'i' -> ABIERTO    | flujo libre del cosmos
Linea  1, Columna  3 | 'e' -> ENCERRADO   | camara sagrada de vacio
Linea  1, Columna  4 | 'n' -> SEMIABIERTO | puerta de vacio
Linea  1, Columna  5 | 'v' -> ABIERTO    | flujo libre del cosmos
Linea  1, Columna  6 | 'e' -> ENCERRADO   | camara sagrada de vacio
Linea  1, Columna  7 | 'n' -> SEMIABIERTO | puerta de vacio
Linea  1, Columna  8 | 'i' -> ABIERTO    | flujo libre del cosmos
Linea  1, Columna  9 | 'd' -> ENCERRADO   | camara sagrada de vacio
Linea  1, Columna 10 | 'o' -> ENCERRADO   | camara sagrada de vacio
Linea  2, Columna  1 | 'a' -> ENCERRADO   | camara sagrada de vacio
```

Línea 2, Columna 3	'H' -> SEMIABIERTO	puerta de vacío
Línea 2, Columna 4	'P' -> ENCERRADO	camara sagrada de vacío
Línea 2, Columna 6	'C' -> SEMIABIERTO	puerta de vacío
Línea 2, Columna 7	'o' -> ENCERRADO	camara sagrada de vacío
Línea 2, Columna 8	'd' -> ENCERRADO	camara sagrada de vacío
Línea 2, Columna 9	'e' -> ENCERRADO	camara sagrada de vacío
Línea 2, Columna 10	'w' -> SEMIABIERTO	puerta de vacío
Línea 2, Columna 11	'a' -> ENCERRADO	camara sagrada de vacío
Línea 2, Columna 12	'r' -> SEMIABIERTO	puerta de vacío
Línea 2, Columna 13	's' -> SEMIABIERTO	puerta de vacío
Línea 3, Columna 1	'2' -> SEMIABIERTO	puerta de vacío
Línea 3, Columna 2	'0' -> ENCERRADO	camara sagrada de vacío
Línea 3, Columna 3	'2' -> SEMIABIERTO	puerta de vacío
Línea 3, Columna 4	'6' -> ENCERRADO	camara sagrada de vacío

Solutions

Python

```
def clasificar_caracter_visual(ch):
```

```
    """
```

Clasifica un carácter según el tipo de espacio VISUAL que suele formar en una tipografía latina estándar, desde la perspectiva ixiana.

Categorías:

- "encerrado" : tiene un hueco interior cerrado
(ej.: a, b, d, e, g, o, p, q, A, B, D, O, P, Q, R, 0, 4, 6, 8, 9)
- "semiabierto" : su forma sugiere uno o varios huecos parcialmente

abiertos

(ej.: c, f, h, k, m, n, r, s, u, y, w,
C, F, H, K, M, N, R, S, U, Y, W, 2, 3, 5)

- "abierto" : no forma hueco interior significativo
(ej.: i, j, l, t, v, x, z, I, J, L, T, V, X, Z, 1, 7)
- "ninguno" : espacios, saltos de línea, signos de puntuación,

etc.

```
    """
```



<CODEWARS2026/>
LEÓN

```
# Conjuntos aproximados, puramente conceptuales
encerrados = set("abdegopqABDOPQR04689")
semiabiertos = set("cfhkmnrsuywCFHKMNRSUYW235")
abiertos = set("ijltvxyzIJLTVXYZ17")
if ch in encerrados:
    return "encerrado"
elif ch in semiabiertos:
    return "semiabierto"
elif ch in abiertos:
    return "abierto"
else:
    # Por simplicidad, dejamos fuera espacios, signos, etc.
    return "ninguno"

def analizar_texto_visual(texto):
    """
    Recorre el texto y analiza cada carácter desde el punto de vista
    visual
    (espacios internos de las letras).
    Devuelve una lista de diccionarios con:
    - indice: posición absoluta en la cadena
    - linea, columna: posición en el texto (comenzando en 1)
    - caracter: el propio carácter
    - categoria: encerrado, semiabierto, abierto o ninguno
    """
    resultados = []
    linea = 1
    columna = 1
    for i, ch in enumerate(texto):
```




<CODEWARS2026/>
LEÓN

```
categoria = clasificar_caracter_visual(ch)
if categoria != "ninguno":
    resultados.append({
        "indice": i,
        "linea": linea,
        "columna": columna,
        "caracter": ch,
        "categoria": categoria
    })
# Actualizar línea y columna
if ch == "\n":
    linea += 1
    columna = 1
else:
    columna += 1
return resultados
def generar_informe_visual(texto):
    """
    Genera y muestra por pantalla un informe alienígena
    sobre los huecos visuales de las letras del texto.
    """
    datos = analizar_texto_visual(texto)
    interpretacion = {
        "encerrado": "camara sagrada de vacio",
        "semiabierto": "puerta de vacio",
        "abierto": "flujo libre del cosmos"
    }
    conteo = {
```



<CODEWARS2026/>
LEÓN

```
"encerrado": 0,
"semiabierto": 0,
"abierto": 0
}
for d in datos:
    conteo[d["categoria"]] += 1
print("--- Resumen visual ---")
print(f"Letras de hueco ENCERRADO    : {conteo['encerrado']}")
print(f"Letras de hueco SEMIABIERTO : {conteo['semiabierto']}")
print(f"Letras de hueco ABIERTO      : {conteo['abierto']}\n")
print("--- Detalle de letras con vacio significativo ---")
for d in datos:
    cat = d["categoria"]
    print(
        f"Linea {d['linea']:>2}, Columna {d['columna']:>2} | "
        f"'{d['caracter']}' -> {cat.upper():11} | "
        {interpretacion[cat]}
    )
if __name__ == "__main__":
    texto_usuario = ""
    while True:
        linea = input()
        if linea.strip() == "": # línea en blanco
            break
        texto_usuario += linea + "\n"
    texto_usuario = texto_usuario.rstrip("\n")
    generar_informe_visual(texto_usuario)
```

C++



<CODEWARS2026/> LEÓN

```
#include <iostream>
#include <string>
#include <vector>
#include <unordered_set>
#include <algorithm>
#include <iomanip>
// ----- Clasificación -----
const std::unordered_set<char> encerrados {
    'a','b','d','e','g','o','p','q',
    'A','B','D','O','P','Q','R',
    '0','4','6','8','9'
};
const std::unordered_set<char> semiabiertos {
    'c','f','h','k','m','n','r','s','u','y','w',
    'C','F','H','K','M','N','R','S','U','Y','W',
    '2','3','5'
};
const std::unordered_set<char> abiertos {
    'i','j','l','t','v','x','z',
    'I','J','L','T','V','X','Z',
    '1','7'
};
std::string clasificar_caracter_visual(char ch) {
    if (encerrados.count(ch)) return "encerrado";
    if (semiabiertos.count(ch)) return "semiabierto";
    if (abiertos.count(ch)) return "abierto";
    return "ninguno";
}
```



<CODEWARS2026/>
LEÓN

```
// ----- Estructura para resultados -----  
struct Registro {  
    int indice;  
    int linea;  
    int columna;  
    char caracter;  
    std::string categoria;  
};  
// ----- Analizar texto -----  
std::vector<Registro> analizar_texto_visual(const std::string &texto) {  
    std::vector<Registro> resultados;  
    int linea = 1;  
    int columna = 1;  
    for (int i = 0; i < (int)texto.size(); i++) {  
        char ch = texto[i];  
        std::string cat = clasificar_caracter_visual(ch);  
        if (cat != "ninguno") {  
            resultados.push_back({i, linea, columna, ch, cat});  
        }  
        if (ch == '\n') {  
            linea++;  
            columna = 1;  
        } else {  
            columna++;  
        }  
    }  
    return resultados;  
}
```



```
// ----- Informe -----  
void generar_informe_visual(const std::string &texto) {  
    auto datos = analizar_texto_visual(texto);  
    int conteo_enc = 0;  
    int conteo_semi = 0;  
    int conteo_ab = 0;  
    for (const auto &d : datos) {  
        if (d.categoria == "encerrado") conteo_enc++;  
        else if (d.categoria == "semiabierto") conteo_semi++;  
        else if (d.categoria == "abierto") conteo_ab++;  
    }  
    std::cout << "--- Resumen visual ---\n";  
    std::cout << "Letras de hueco ENCERRADO    : " << conteo_enc << "\n";  
    std::cout << "Letras de hueco SEMIABIERTO : " << conteo_semi << "\n";  
    std::cout << "Letras de hueco ABIERTO      : " << conteo_ab << "\n\n";  
    std::cout << "--- Detalle de letras con vacio significativo ---\n";  
    for (const auto &d : datos) {  
        std::string interpretacion =  
            (d.categoria == "encerrado")    ? "camara sagrada de vacio" :  
            (d.categoria == "semiabierto") ? "puerta de vacio" :  
                                                "flujo libre del cosmos";  
        std::string categoriaMayus = d.categoria;  
        std::transform(categoriaMayus.begin(), categoriaMayus.end(),  
            categoriaMayus.begin(),  
                [](unsigned char c){ return std::toupper(c); });  
        std::cout << "Linea " << std::setw(2) << std::right << d.linea  
            << ", Columna " << std::setw(2) << std::right <<  
            d.columna
```

```
        << " | '" << d.caracter << "' -> "
        << std::left << std::setw(11) << categoriaMayus // <-- aqui
el padding
        << " | " << interpretacion
        << "\n";
    }
}
// ----- Programa principal -----
int main() {
    std::string texto_usuario;
    std::string linea;
    while (true) {
        if (!std::getline(std::cin, linea))
            break;
        // Linea en blanco -> salir
        std::string trimmed = linea;
        trimmed.erase(0, trimmed.find_first_not_of(" \t"));
        if (trimmed.empty()) break;
        texto_usuario += linea + "\n";
    }
    // Eliminar salto final si existe
    if (!texto_usuario.empty() && texto_usuario.back() == '\n')
        texto_usuario.pop_back();
    generar_informe_visual(texto_usuario);
    return 0;
}
```

Java

```
import java.util.*;
```



```
public class cartography {  
    // ----- Clasificación -----  
    private static final Set<Character> ENCERRADOS = new HashSet<>(  
        Arrays.asList(  
            'a','b','d','e','g','o','p','q',  
            'A','B','D','O','P','Q','R',  
            '0','4','6','8','9'  
        )  
    );  
    private static final Set<Character> SEMIABIERTOS = new HashSet<>(  
        Arrays.asList(  
            'c','f','h','k','m','n','r','s','u','y','w',  
            'C','F','H','K','M','N','R','S','U','Y','W',  
            '2','3','5'  
        )  
    );  
    private static final Set<Character> ABIERTOS = new HashSet<>(  
        Arrays.asList(  
            'i','j','l','t','v','x','z',  
            'I','J','L','T','V','X','Z',  
            '1','7'  
        )  
    );  
    public static String clasificarCaracter(char ch) {  
        if (ENCERRADOS.contains(ch)) return "encerrado";  
        if (SEMIABIERTOS.contains(ch)) return "semiabierto";  
        if (ABIERTOS.contains(ch)) return "abierto";  
        return "none";  
    }  
}
```



<CODEWARS2026/>
LEÓN

```
}  
// ----- Estructura registro -----  
public static class Registro {  
    int indice;  
    int linea;  
    int columna;  
    char character;  
    String categoria;  
    Registro(int indice, int linea, int columna, char character, String  
categoria) {  
        this.indice = indice;  
        this.linea = linea;  
        this.columna = columna;  
        this.character = character;  
        this.categoria = categoria;  
    }  
}  
// ----- Analizar texto -----  
public static List<Registro> analizarTexto(String texto) {  
    List<Registro> resultados = new ArrayList<>();  
    int linea = 1;  
    int columna = 1;  
    for (int i = 0; i < texto.length(); i++) {  
        char ch = texto.charAt(i);  
        String categoria = clasificarCaracter(ch);  
        if (!categoria.equals("none")) {  
            resultados.add(new Registro(i, linea, columna, ch,  
categoria));  
        }  
    }  
}
```




<CODEWARS2026/>
LEÓN

```
    }
    if (ch == '\n') {
        linea++;
        columna = 1;
    } else {
        columna++;
    }
}
return resultados;
}

// ----- Informe visual -----
public static void generarInforme(String texto) {
    List<Registro> datos = analizarTexto(texto);
    int encerrado = 0;
    int semiabierto = 0;
    int abierto = 0;
    for (Registro r : datos) {
        switch (r.categoria) {
            case "encerrado": encerrado++; break;
            case "semiabierto": semiabierto++; break;
            case "abierto": abierto++; break;
        }
    }
    System.out.println("--- Resumen visual ---");
    System.out.println("Letras de hueco ENCERRADO    : " + encerrado);
    System.out.println("Letras de hueco SEMIABIERTO : " +
semiabierto);
    System.out.println("Letras de hueco ABIERTO      : " + abierto);
}
```



<CODEWARS2026/>
LEÓN

```
System.out.println();
System.out.println("--- Detalle de letras con vacio significativo
---");
for (Registro r : datos) {
    String interpretacion =
        r.categoria.equals("encerrado") ? "camara sagrada de
vacio" :
        r.categoria.equals("semiabierto") ? "puerta de vacio" :
        "flujo libre del
cosmos";

    System.out.printf(
        "Linea %2d, Columna %2d | '%c' -> %-11s | %s%n",
        r.linea, r.columna, r.caracter, r.categoria.toUpperCase(),
        interpretacion
    );
}
}
// ----- Main -----
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    StringBuilder sb = new StringBuilder();
    while (true) {
        if (!sc.hasNextLine()) break;
        String linea = sc.nextLine();
        if (linea.trim().isEmpty()) break;
        sb.append(linea).append("\n");
    }
    // Eliminar salto final
```



<CODEWARS2026/>
LEÓN

```
String texto = sb.toString();  
if (texto.endsWith("\n")) {  
    texto = texto.substring(0, texto.length() - 1);  
}  
generarInforme(texto);  
}  
}
```

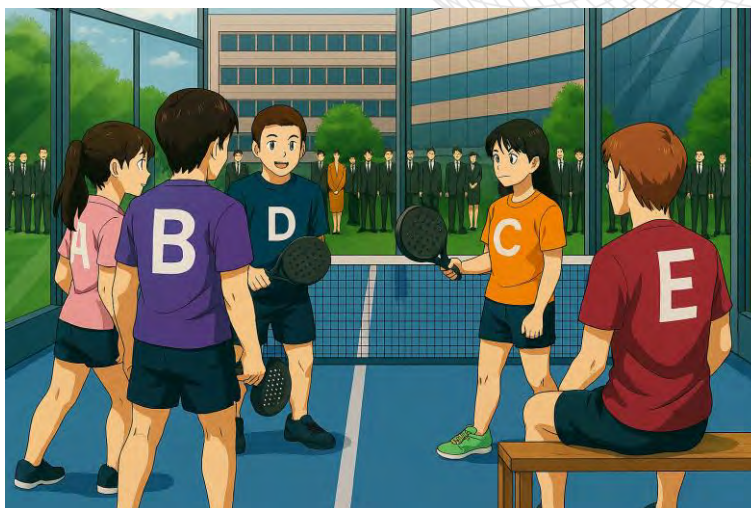
<let's code/>



17

Padel Ranking Calculator

10 points



Introduction

An IT company is organizing a padel tournament among its employees. Instead of always playing with the same pairs, the company wants employees to interact with each other and the padel ranking to be individual. Therefore, all matches resulting from the combinations of players must be played.

For example, if 5 employees sign up to the tournament (player A, B, C, D and E), player A will play with player B against all combinations of players one match. Therefore the pair AB will play against CD, against CE and against DE. An so on with all other combinations. It will, therefore, be necessary to play the following matches:

AB vs CD
AB vs DE
AB vs CE


```
AC vs BD
AC vs BE
AC vs DE
AD vs BC
AD vs BE
AD vs CE
AE vs BC
AE vs BD
AE vs CD
BC vs DE
BD vs CE
BE vs CD
```

To simplify this exercise, we'll assume that ALWAYS the tournaments have 5 players and the players are A, B, C, D, and E.

Exercise

You are asked to write a program that reads the results of the 15 matches and outputs the individual final ranking. The ranking will be ordered by the number of matches won. In case of a tie, the number of sets won will decide the winner; if there is still a tie, the number of games won and, in the extreme there is still a tie, the letter order won.

Input

The program will receive 15 lines with match results in a format like the following (please note the spacing):

```
AE vs BC:7-6, 5-7, 6-3
AB vs CE:5-7, 6-3, 7-6
AB vs DE:
AC vs BD:7-6, 5-7, 6-3
...
BE vs CD:
```

The input order can be any, but all 15 matches of the tournament will be present. Additionally, it might be possible that a match has not been played. In the previous example, match 3 has not been played.

Output

The output must be formatted as follows:

```
1- Player C - 11 Matches Won - 24 Sets Won - 135 Games Won
2- Player A - 11 Matches Won - 23 Sets Won - 139 Games Won
3- Player B - 10 Matches Won - 25 Sets Won - 119 Games Won
...
```

Example 1

Input

```
AB vs CD:7-6, 5-7, 6-3
AB vs CE:5-7, 6-3, 7-6
AB vs DE:
AC vs BD:7-6, 5-7, 6-3
AC vs BE:5-7, 6-3, 7-6
AC vs DE:
AD vs BC:7-6, 5-7, 6-3
AD vs BE:
AD vs CE:
AE vs BC:
AE vs BD:
AE vs CD:
BC vs DE:6-0, 6-1, 6-2
BD vs CE:
BE vs CD:0-6, 0-6, 0-6
```

Output

```
1- Player A - 5 Matches Won - 10 Sets Won - 90 Games Won
2- Player C - 4 Matches Won - 13 Sets Won - 120 Games Won
3- Player B - 3 Matches Won - 10 Sets Won - 102 Games Won
4- Player D - 2 Matches Won - 7 Sets Won - 71 Games Won
5- Player E - 0 Matches Won - 2 Sets Won - 35 Games Won
```

Example 2

Input

```
AB vs CD:0-6, 5-7, 6-3
AB vs CE:5-7, 6-3, 7-6
```

```
AB vs DE:1-6, 2-6, 6-0
AC vs BD:7-6, 5-7, 6-3
AC vs BE:5-7, 6-3, 7-6
AC vs DE:6-7, 6-7, 6-7
AD vs BC:7-6, 5-7, 6-3
AD vs BE:6-7, 6-7, 6-7
AD vs CE:6-0, 6-0, 6-0
AE vs BC:5-7, 5-7, 5-7
AE vs BD:
AE vs CD:
BC vs DE:6-0, 6-1, 6-2
BD vs CE:
BE vs CD:0-6, 0-6, 0-6
```

Output

```
1- Player C - 6 Matches Won - 17 Sets Won - 159 Games Won
2- Player D - 6 Matches Won - 16 Sets Won - 140 Games Won
3- Player A - 5 Matches Won - 13 Sets Won - 161 Games Won
4- Player B - 4 Matches Won - 16 Sets Won - 146 Games Won
5- Player E - 3 Matches Won - 10 Sets Won - 104 Games Won
```

Solutions

Python

```
def parse_result(line):
    # Example: AE vs BC:7-6, 5-7, 6-3
    teams, sets = line.strip().split(":")
    team1, team2 = teams.split(" vs ")
    sets = [tuple(map(int, s.split("-"))) for s in sets.split(",")]
    return team1, team2, sets

def update_stats(player, match_won, sets_won, games_won, stats):
    stats[player]['matches'] += match_won
    stats[player]['sets'] += sets_won
    stats[player]['games'] += games_won

def main():
    players = set("ABCDE")
```



<CODEWARS2026/>
LEÓN

```
stats = {p: {'matches': 0, 'sets': 0, 'games': 0} for p in players}
for _ in range(15):
    line = input()
    _, sets = line.strip().split(":")
    if sets == "":
        continue
    team1, team2, sets = parse_result(line)
    # Validate players.
    for player in team1 + team2:
        if player not in players:
            raise ValueError("Invalid player: {}".format(player))
    # Count sets and games.
    team1_sets = team2_sets = 0
    team1_games = team2_games = 0
    for s1, s2 in sets:
        if s1 == s2 or s1 > 7 or s2 > 7 or s1 < 0 or s2 < 0:
            raise ValueError("Invalid set score: {}-{}".format(s1,
s2))

        if s1 > s2:
            team1_sets += 1
        else:
            team2_sets += 1
        team1_games += s1
        team2_games += s2
    # Determine match winner and update stats.
    winner = team1 if team1_sets > team2_sets else team2
    results = {team1: (team1 == winner, team1_sets, team1_games),
team2: (team2 == winner, team2_sets, team2_games)}
```




<CODEWARS2026/>
LEÓN

```
    for team, victories in results.items():
        match_won, sets_won, games_won = victories
        for player in team:
            update_stats(player, match_won, sets_won, games_won,
stats)
    # Sort and display ranking.
    ranking = sorted(stats.items(), key=lambda x: (-x[1]['matches'], -
x[1]['sets'], -x[1]['games'], x[0]))
    for pos, (player, data) in enumerate(ranking, 1):
        print(f"{pos}- Player {player} - {data['matches']} Matches Won -
{data['sets']} Sets Won - {data['games']} Games Won")
if __name__ == "__main__":
    main()
```

C++

```
#include <iostream>
#include <map>
#include <vector>
#include <algorithm>
#include <set>
#include <sstream>
struct Stats {
    int matches = 0, sets = 0, games = 0;
};
void updateStats(char player, int matchWon, int setsWon, int gamesWon,
std::map<char, Stats>& stats) {
    stats[player].matches += matchWon;
    stats[player].sets += setsWon;
    stats[player].games += gamesWon;
```



<CODEWARS2026/>
LEÓN

```
}  
int main() {  
    std::set<char> players = {'A','B','C','D','E'};  
    std::map<char, Stats> stats;  
    for (char player : players) stats[player] = Stats();  
    for (int i = 0; i < 15; ++i) {  
        std::string line;  
        std::getline(std::cin, line);  
        if (line.find(':') == std::string::npos || line.back() == ':')  
            continue;  
        std::string teams = line.substr(0, line.find(':'));  
        std::string setsStr = line.substr(line.find(':') + 1);  
        std::string team1 = teams.substr(0, teams.find(" vs "));  
        std::string team2 = teams.substr(teams.find(" vs ") + 4);  
        // Validate players.  
        for (char player : team1 + team2) {  
            if (players.find(player) == players.end()) {  
                throw std::invalid_argument("Invalid player: " +  
std::string(1, player));  
            }  
        }  
        // Count sets and games  
        int team1Sets = 0, team2Sets = 0, team1Games = 0, team2Games = 0;  
        std::stringstream ss(setsStr);  
        std::string set;  
        while (std::getline(ss, set, ',')) {  
            if (set.empty()) continue;  
            int s1, s2;
```

```
        sscanf(set.c_str(), "%d-%d", &s1, &s2);
        if (s1 == s2 || s1 > 7 || s2 > 7 || s1 < 0 || s2 < 0) {
            throw std::invalid_argument("Invalid set score: " +
std::to_string(s1) + "-" + std::to_string(s2));
        }
        if (s1 > s2) team1Sets++; else team2Sets++;
        team1Games += s1;
        team2Games += s2;
    }
    // Determine match winner and update stats
    std::string winner = (team1Sets > team2Sets) ? team1 : team2;
    std::vector<std::pair<std::string, std::tuple<bool, int, int>>>
results = {
    {team1, {team1 == winner, team1Sets, team1Games}},
    {team2, {team2 == winner, team2Sets, team2Games}}
};
    for (const auto& [team, victories] : results) {
        auto [won, sets_won, games_won] = victories;
        for (char player : team) {
            updateStats(player, won ? 1 : 0, sets_won, games_won,
stats);
        }
    }
    // Sort and display ranking.
    std::vector<std::pair<char, Stats>> ranking(stats.begin(),
stats.end());
```



<CODEWARS2026/>
LEÓN

```
std::sort(ranking.begin(), ranking.end(), [](const auto& a, const
auto& b) {
    if (b.second.matches != a.second.matches)
        return b.second.matches < a.second.matches;
    if (b.second.sets != a.second.sets)
        return b.second.sets < a.second.sets;
    if (b.second.games != a.second.games)
        return b.second.games < a.second.games;
    return a.first < b.first;
});
int pos = 1;
for (const auto& entry : ranking) {
    const Stats& d = entry.second;
    printf("%d- Player %c - %d Matches Won - %d Sets Won - %d Games
Won\n",
        pos++, entry.first, d.matches, d.sets, d.games);
}
return 0;
}
```

Java

```
import java.util.*;
public class PadelRanking {
    static class Stats {
        int matches = 0, sets = 0, games = 0;
    }
    static void updateStats(char player, int matchWon, int setsWon, int
gamesWon, Map<Character, Stats> stats) {
        Stats s = stats.get(player);
```

<let's code/>





<CODEWARS2026/>
LEÓN

```
s.matches += matchWon;
s.sets += setsWon;
s.games += gamesWon;
}
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    Set<Character> players = new
HashSet<>(Arrays.asList('A','B','C','D','E'));
    Map<Character, Stats> stats = new HashMap<>();
    for (char player : players) stats.put(player, new Stats());
    for (int i = 0; i < 15; i++) {
        String line = sc.nextLine().trim();
        if (!line.contains(":") || line.endsWith(":")) continue;
        String[] parts = line.split(":");
        String[] teams = parts[0].split(" vs ");
        String team1 = teams[0], team2 = teams[1];
        // Validate players.
        String allPlayers = team1 + team2;
        for (char player : allPlayers.toCharArray()) {
            if (!players.contains(player)) {
                throw new IllegalArgumentException("Invalid player: "
+ player);
            }
        }
        // Count sets and games.
        String[] setsStr = parts[1].split(",");
        int team1Sets = 0, team2Sets = 0, team1Games = 0, team2Games =
0;
```



<CODEWARS2026/>
LEÓN

```
for (String set : setsStr) {
    set = set.trim();
    if (set.isEmpty()) continue;
    String[] gamesWon = set.split("-");
    int s1 = Integer.parseInt(gamesWon[0]);
    int s2 = Integer.parseInt(gamesWon[1]);
    if (s1 == s2 || s1 > 7 || s2 > 7 || s1 < 0 || s2 < 0) {
        throw new IllegalArgumentException("Invalid set score:
" + s1 + "-" + s2);
    }
    if (s1 > s2) team1Sets++; else team2Sets++;
    team1Games += s1;
    team2Games += s2;
}

// Determine match winner and update stats.
String winner = (team1Sets > team2Sets) ? team1 : team2;
Map<String, int[]> results = new HashMap<>();
results.put(team1, new int[]{team1.equals(winner) ? 1 : 0,
team1Sets, team1Games});
results.put(team2, new int[]{team2.equals(winner) ? 1 : 0,
team2Sets, team2Games});
for (Map.Entry<String, int[]> entry : results.entrySet()) {
    String team = entry.getKey();
    int[] victories = entry.getValue();
    int matchWon = victories[0];
    int setsWon = victories[1];
    int gamesWon = victories[2];
    for (char player : team.toCharArray()) {
```

```
        updateStats(player, matchWon, setsWon, gamesWon,
stats);
    }
}
// Sort and display ranking.
List<Map.Entry<Character, Stats>> ranking = new
ArrayList<>(stats.entrySet());
ranking.sort((a, b) -> {
    if (b.getValue().matches != a.getValue().matches)
        return b.getValue().matches - a.getValue().matches;
    if (b.getValue().sets != a.getValue().sets)
        return b.getValue().sets - a.getValue().sets;
    if (b.getValue().games != a.getValue().games)
        return b.getValue().games - a.getValue().games;
    return a.getKey() - b.getKey();
});
int pos = 1;
for (Map.Entry<Character, Stats> entry : ranking) {
    Stats d = entry.getValue();
    System.out.printf("%d- Player %c - %d Matches Won - %d Sets
Won - %d Games Won\n",
        pos++, entry.getKey(), d.matches, d.sets, d.games);
}
}
```


18

Roland Garros Point Counter

11 points

Introduction

You are still working on the Roland Garros project, and it turns out that the person in charge of programming the match scoreboard has left the company, leaving their part of the project unfinished.

Since you did a great job with the Tie-Break scoring, you have been assigned the development of the match scoreboard.



As with the Tie-Break program, the AI analyzes the game and detects who won each point. And as we all know, computers work with 0s and 1s, so the AI simply sends us zeros and ones.

Specifically, it sends us a string of zeros and ones separated by commas, just like in the Tie-Break, where 0 means that “Player 1” won the point and 1 means that “Player 2” won the point. Careful, don’t get confused: a “1” does NOT mean Player 1 won the point.

The scoring in a game is 0, 15, 30, 40. If it reaches 40-40, the player who wins the next point has “advantage,” which we will call “Adv.” If the player with advantage loses the point, it goes back to 40-40. But if they win it, they win the game. A “set” is won when a player reaches 6 games, provided they have a two-game lead. For example, a set can be won if a player wins the sixth game and the score is 6-4. But if it's 6-5, the set continues and could end 7-5 or 6-6. If it ends 6-6, that's when the Tie-Break is played. Since you already programmed that part in a previous exercise, and to simplify, we will not address Tie-Breaks here. That means if a set ends 6-6, it will appear as such on the scoreboard.

Exercise

You are asked to develop a program that receives a list of zeros and ones corresponding to Players 1 and 2 and tells us the current match score.

Note that the program is still in the testing phase and has a couple of issues:

- Points played during Tie-Breaks are not counted. If a game ends in a Tie-Break, we will show it as 6-6 and not add a winner (see example 3).
- The program does not yet detect the end of the match; more than 5 sets could appear.

Input

The input will be a list of zeros and ones. For example:

```
1,1,0,1,1,0,1,0,0,0,1,1,0,0,0,1,1,0,0,0,0,1,1,0,1,0,1,0,1,1,1,0,0,0,1,1,0,1,1,0,1,1,0,1,1,0,1,0,0,0,1,1,0,0,0,1,1,0,0,0,0,1,1,0,0,0,0,1,1,0,1,0,1,0,1,1,1,0,0,0,1,1,0
```

Output

The program will generate an output with the following format (in spanish):

Jugador 1: <sets_ganados> <Juegos_ganados_set_actual> <Puntos_juego_actual>

Jugador 2: <sets_ganados> <Juegos_ganados_set_actual> <Puntos_juego_actual>

Sets terminados <set1>,<set2>,<set3>

For instance, with the given input:

```
Jugador 1: 0 4 Adv  
Jugador 2: 0 4 40  
Sets terminados:
```

Note that even when there are no "Set terminados", there's an space AFTER THE COLON.

Example1

Input

```
1,1,0,1,1,0,1,0,0,0,1,1,0,0,0,1,1,0,0,0,0,1,1,0,1,0,1,0,1,1,1,0,0,0,1,1,0,1,1,0,1,  
1,0,1,0,0,0,1,1,0,0,0,1,1,0,0,0,0,1,1,0,1,0,1,0,1,1,1,0,0,0,1,1,0,0,0,1,1,0
```

Output

```
Jugador 1: 0 4 Adv  
Jugador 2: 0 4 40  
Sets terminados:
```

Example2

Input

```
1,1,0,1,1,0,1,0,0,0,1,1,0,0,0,1,1,0,0,0,0,1,1,0,1,0,1,0,1,1,1,0,0,0,1,1,0,1,1,0,1,  
1,0,1,0,0,0,1,1,0,0,0,1,1,0,0,0,0,1,1,0,1,0,1,0,1,1,1,0,0,0,1,1,0,1,1,1,0,0,0,1,1,  
0,0,0,0,1,1,0,1,0,1,0,1,1,1,0,0,0,1,1,0,1,1,0,1,1,0,1,1,0,1,0,0,0,1,1,0,0,0,1,1,0,  
1,1,0,1,0,1,0,1,1,1,0,0,0,1,1,0,1,1,0,1,1,0,1,0,0,0,1,1,0,0,0,1,1,0,0,1,1,0,1,  
0,1,0,1,1,1,0,0,0
```

Output

```
Jugador 1: 0 4 40  
Jugador 2: 0 5 15  
Sets terminados: 6-6
```

Example3

Input

```
1,1,0,1,1,0,1,0,0,0,1,1,0,0,0,1,1,0,0,0,0,1,1,0,1,0,1,0,1,1,1,0,0,0,1,1,0,1,1,0,1,
1,0,1,0,0,0,1,1,0,0,0,1,1,0,0,0,0,1,1,0,1,0,1,0,1,1,1,0,0,0,1,1,0,1,1,0,0,0,1,1,
0,0,0,0,1,1,0,1,0,1,0,1,1,1,0,0,0,1,1,0,1,1,0,1,1,0,1,1,0,0,0,1,1,0,0,0,1,1,0,
1,1,0,1,0,1,0,1,1,1,0,0,0,1,1,0,1,1,0,1,1,0,1,1,0,1,0,0,0,1,1,0,0,0,1,1,0,
0,1,0,1,1,1,0,0,0,1,1,1,1,0,0,0,1,1,1,1,0,0,0,1,1,0,0,0,0,1,1,0,1,
0,1,0,1,1,1,0,0,0,0,1,1,1,1,1,0,0,0,0
```

Output

```
Jugador 1: 0 0 15
Jugador 2: 0 0 0
Sets terminados: 6-6,6-6
```

Solutions

Python

global ventaja

```
def marcador_tenis(puntos):
    puntuacion = [0, 15, 30, 40]
    sets = [0, 0]      # Sets ganados por cada jugador
    juegos = [0, 0]    # Juegos en el set actual
    puntos_actuales = [0, 0]
    ventaja = None
    sets_terminados = [] # Lista para guardar sets finalizados
    for punto in puntos:
        #ganador y perdedor son 0 o 1
        ganador = punto
        perdedor = 1 - ganador
        # Si hay ventaja
        if ventaja is not None:
            if ganador == ventaja:
```

```
juegos[ganador] += 1
ventaja = None
puntos_actuales = [0, 0]
else:
    ventaja = None
    puntos_actuales[ganador] += 1
else:
    puntos_actuales[ganador] += 1
    #print (str(puntos_actuales[0])+"-"+str(puntos_actuales[1]))
    #1 --> 15, 2--> 30, 3 --> 40 , 4--> Juego o Adv. Se puede
    ganar con 4, si el perdedor solo tiene 2 (30). 0 con 5, si el
    #Perdedor solo tiene 3 (40)
    if puntos_actuales[ganador] >= 4 and puntos_actuales[ganador]
- puntos_actuales[perdedor] >= 2:
        juegos[ganador] += 1
        #Reseteamos los puntos del juego
        puntos_actuales = [0, 0]
        #Estamos en 40-40 o en Adv-40
        elif puntos_actuales[0] >= 3 and puntos_actuales[1] >= 3:
            if puntos_actuales[ganador] > puntos_actuales[perdedor]:
                ventaja = ganador

    #print (str(punto)+"--> Estamos "+str(juegos[0])+"-
"+str(juegos[1])+" y "+(str(puntos_actuales[0])+"-"+
str(puntos_actuales[1]) ))
    # Comprobar si alguien gana el set
    if juegos[ganador] >= 6 and juegos[ganador] - juegos[perdedor] >=
2:

        sets[ganador] += 1
```



```
sets_terminados.append(f"{juegos[0]}-{juegos[1]}")
juegos = [0, 0]
puntos_actuales = [0, 0]
ventaja = None
elif juegos[0] == 6 and juegos[1] == 6:
    # Set termina en 6-6 (sin tie-break)
    sets_terminados.append("6-6")
    juegos = [0, 0]
    puntos_actuales = [0, 0]
    ventaja = None
# Determinar puntos actuales para mostrar
def puntos_texto(ventaja,puntos_actuales):
    if ventaja is not None:
        return ("Adv" if ventaja == 0 else "40", "Adv" if ventaja == 1
else "40")
    else:
        return (
            puntuacion[min(puntos_actuales[0], 3)],
            puntuacion[min(puntos_actuales[1], 3)]
        )
p1, p2 = puntos_texto(ventaja, puntos_actuales)
# Mostrar marcador
print(f"Jugador 1: {sets[0]} {juegos[0]} {p1}")
print(f"Jugador 2: {sets[1]} {juegos[1]} {p2}")
# Mostrar sets terminados
print("Sets terminados: " + ",".join(sets_terminados))
# Leer entrada
entrada = input().strip()
```



<CODEWARS2026/>
LEÓN

```
puntos = [int(x) for x in entrada.split(",")]  
marcador_tenis(puntos)
```

C++

```
#include <iostream>  
#include <vector>  
#include <sstream>  
#include <string>  
using namespace std;  
void marcadorTennis(const vector<int>& puntos) {  
    int puntuacion[] = {0, 15, 30, 40};  
    int sets[2] = {0, 0};  
    int juegos[2] = {0, 0};  
    int puntosActuales[2] = {0, 0};  
    int ventaja = -1;  
    vector<string> setsTerminados;  
    for (int punto : puntos) {  
        int ganador = punto;  
        int perdedor = 1 - ganador;  
        if (ventaja != -1) {  
            if (ganador == ventaja) {  
                juegos[ganador]++;  
                ventaja = -1;  
                puntosActuales[0] = puntosActuales[1] = 0;  
            } else {  
                ventaja = -1;  
                puntosActuales[ganador]++;  
            }  
        } else {  

```

<let's code/>



```
puntosActuales[ganador]++;  
if (puntosActuales[ganador] >= 4 && puntosActuales[ganador] -  
puntosActuales[perdedor] >= 2) {  
    juegos[ganador]++;  
    puntosActuales[0] = puntosActuales[1] = 0;  
} else if (puntosActuales[0] >= 3 && puntosActuales[1] >= 3) {  
    if (puntosActuales[ganador] > puntosActuales[perdedor]) {  
        ventaja = ganador;  
    }  
}  
}  
if (juegos[ganador] >= 6 && juegos[ganador] - juegos[perdedor] >=  
2) {  
    sets[ganador]++;  
    setsTerminados.push_back(to_string(juegos[0]) + "-" +  
to_string(juegos[1]));  
    juegos[0] = juegos[1] = 0;  
    puntosActuales[0] = puntosActuales[1] = 0;  
    ventaja = -1;  
} else if (juegos[0] == 6 && juegos[1] == 6) {  
    setsTerminados.push_back("6-6");  
    juegos[0] = juegos[1] = 0;  
    puntosActuales[0] = puntosActuales[1] = 0;  
    ventaja = -1;  
}  
}  
string p1, p2;  
if (ventaja != -1) {
```



<CODEWARS2026/>
LEÓN

```
        p1 = (ventaja == 0) ? "Adv" : "40";
        p2 = (ventaja == 1) ? "Adv" : "40";
    } else {
        p1 = to_string(puntuacion[min(puntosActuales[0], 3)]);
        p2 = to_string(puntuacion[min(puntosActuales[1], 3)]);
    }
    cout << "Jugador 1: " << sets[0] << " " << juegos[0] << " " << p1 <<
endl;
    cout << "Jugador 2: " << sets[1] << " " << juegos[1] << " " << p2 <<
endl;
    cout << "Sets terminados: ";
    for (size_t i = 0; i < setsTerminados.size(); i++) {
        cout << setsTerminados[i];
        if (i < setsTerminados.size() - 1) cout << ",";
    }
    cout << endl;
}

int main() {
    string entrada;
    getline(cin, entrada);
    stringstream ss(entrada);
    string token;
    vector<int> puntos;
    while (getline(ss, token, ',')) {
        puntos.push_back(stoi(token));
    }
    marcadorTenis(puntos);
    return 0;
}
```




<CODEWARS2026/>
LEÓN

```
}
```

Java

```
import java.util.*;

public class RolandGarrosPointCounter {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String entrada = sc.nextLine().trim();
        String[] partes = entrada.split(",");
        int[] puntos = new int[partes.length];
        for (int i = 0; i < partes.length; i++) {
            puntos[i] = Integer.parseInt(partes[i].trim());
        }
        marcadorTenis(puntos);
    }

    public static void marcadorTenis(int[] puntos) {
        int[] puntuacion = {0, 15, 30, 40};
        int[] sets = {0, 0};
        int[] juegos = {0, 0};
        int[] puntosActuales = {0, 0};
        Integer ventaja = null;
        List<String> setsTerminados = new ArrayList<>();
        for (int punto : puntos) {
            int ganador = punto;
            int perdedor = 1 - ganador;
            if (ventaja != null) {
                if (ganador == ventaja) {
                    juegos[ganador]++;
                    ventaja = null;
                }
            }
        }
    }
}
```

<let's code/>



```
        puntosActuales[0] = puntosActuales[1] = 0;
    } else {
        ventaja = null;
        puntosActuales[ganador]++;
    }
} else {
    puntosActuales[ganador]++;
    if (puntosActuales[ganador] >= 4 &&
puntosActuales[ganador] - puntosActuales[perdedor] >= 2) {
        juegos[ganador]++;
        puntosActuales[0] = puntosActuales[1] = 0;
    } else if (puntosActuales[0] >= 3 && puntosActuales[1] >=
3) {
        if (puntosActuales[ganador] >
puntosActuales[perdedor]) {
            ventaja = ganador;
        }
    }
}
if (juegos[ganador] >= 6 && juegos[ganador] - juegos[perdedor]
>= 2) {
    sets[ganador]++;
    setsTerminados.add(juegos[0] + "-" + juegos[1]);
    juegos[0] = juegos[1] = 0;
    puntosActuales[0] = puntosActuales[1] = 0;
    ventaja = null;
} else if (juegos[0] == 6 && juegos[1] == 6) {
    setsTerminados.add("6-6");
}
```

```
juegos[0] = juegos[1] = 0;
puntosActuales[0] = puntosActuales[1] = 0;
ventaja = null;
    }
}
String p1, p2;
if (ventaja != null) {
    p1 = (ventaja == 0) ? "Adv" : "40";
    p2 = (ventaja == 1) ? "Adv" : "40";
} else {
    p1 = String.valueOf(puntuacion[Math.min(puntosActuales[0],
3))]);
    p2 = String.valueOf(puntuacion[Math.min(puntosActuales[1],
3))]);
}
System.out.println("Jugador 1: " + sets[0] + " " + juegos[0] + "
" + p1);
System.out.println("Jugador 2: " + sets[1] + " " + juegos[1] + "
" + p2);
System.out.println("Sets terminados: " + String.join(",",
setsTerminados));
}
}
```


19

Un problema universal

11 points

Introduction

Cuando los Tricachfin descubrieron a los habitantes del planeta de las tres lunas, la galaxia se estremeció. No por el hallazgo –había millones de mundos primitivos–, sino porque decidieron hablarles, saltándose el Protocolo de Silencio Cósmico.

El Protocolo de Silencio Cósmico lo prohibía con absoluta claridad:

“Ninguna mente que controle la energía de un universo debe alterar el destino de una mente que aún no controla la energía de su propio hogar.”

...Pero la curiosidad es un instinto más antiguo que las leyes.

El equipo de enlace –formado por entidades sin forma, que viajaban entre dimensiones del pensamiento– descendió sobre el pequeño mundo azul-verde. Allí ,habitaban los N’mari, una especie que apenas había aprendido a lanzar fragmentos de sí mismos al vacío del espacio.



El problema apareció pronto. Los N'mari no entendían las ecuaciones. No porque fueran simples, sino porque sus matemáticas eran un arte, no una ciencia. Sus sacerdotes pregonaban que sumar y multiplicar eran actos sagrados, gestos de expansión, mientras que restar y dividir eran rituales de destrucción. En su escritura numérica, el orden de los signos dependía del sentimiento: si una ecuación "lucía fea", debía rehacerse hasta que resultara hermosa.

Los Tricachfin intentaron enseñarles el lenguaje universal: las matemáticas puras. Pero cada fórmula enviada a su red de cómputo planetario era reinterpretada con lógica estética. La ecuación de propulsión cuántica, por ejemplo, se transformó en un poema sobre el fuego y el vacío. El modelo de curvatura gravitatoria terminó como una plegaria a sus dioses binarios. En resumen, era como intentar enseñarles matemáticas puras a una gallina.

Así que decidieron buscar un traductor entre civilizaciones: Tú.

Exercise

Como traductor entre ambas civilizaciones, tu misión es construir un intérprete que evalúe expresiones matemáticas tal como lo haría la civilización de los N'mari. Tu tarea es escribir un programa que, dada una cadena con una operación, devuelva su resultado según las reglas estéticas de su mundo.

Reglas de la aritmética estética

- 1.- Los operadores $+$ y $*$ poseen mayor jerarquía: se calculan antes que $-$ y $/$.
- 2.- Las operaciones del mismo grupo se resuelven de izquierda a derecha.
- 3.- No existen paréntesis (son considerados herejía simbólica).
- 4.- Se admiten números enteros o decimales, y signos unarios ($+$ o $-$ al inicio).

La civilización de los N'mari aborrecen las divisiones por cero –ya esto causaría una catástrofe galáctica (ZeroDivisionError). Y si la expresión es ilegible o contradictoria, lanza un ValueError, pues incluso en su universo, el sinsentido no tiene cabida.

Input

El programa recibirá una única línea con una fórmula matemática que cumpla las reglas anteriores. Por ejemplo:

```
3+4*2
```

Output

El programa mostrará el resultado según las reglas de la aritmética estética. Por ejemplo, con la anterior entrada, el resultado en matemática estándar será: 11. Pero para la Civilización N'mari, primero van la suma y la multiplicación (+ y *) y dentro de ellas, el orden (de izquierda a derecha): $3+4=7$, luego $7*2=14$.

El programa simplemente mostrará el resultado:

```
14
```

Nota: Por simplicidad, solo se trabajará con decimales de un único dígito. Los números enteros se mostrarán sin decimales.

Nota 2: Podría ocurrir que la entrada simplemente reciba un número sin ningún tipo de operación, la salida será ese mismo número

Nota 3: Si en algún momento se produce una división por cero, se mostrará por pantalla "ZeroDivisionError"

Nota 4: Y si se produce una entrada inválida: "3++4", "2a+1", "3*(4+2)", se escribirá "ValueError"... recordemos que no hay paréntesis en la civilización N'mari



<CODEWARS2026/> LEÓN

Example 1

Input

8-2/2

Output

3

Ya que no hay + ni *, por la estética N'mari se resuelven - y / al mismo nivel, de izquierda a derecha: $8-2=6$, $6/2=3$

Example 2

Input

10/2+3*2-4

Output

-3

Example 3

Input

-3 + 4 * 2

Output

2

Example 4

Input

2.5*2+1.5



<CODEWARS2026/> LEÓN

Output

6.5

Example 5

Input

42

Output

42

Example 6

Input

1+2/0

Output

ZeroDivisionError

Example 7

Input

3+(4-45)

Output

ValueError

Solutions

Python



<CODEWARS2026/>
LEÓN

```
import re
from typing import Union, List
Number = Union[int, float]
class Token:
    def __init__(self, kind: str, value: str):
        self.kind = kind # 'num' or 'op'
        self.value = value
    def __repr__(self):
        return f"Token({self.kind!r}, {self.value!r})"
def _to_number(s: str) -> Number:
    # Devuelve int si es exacto, si no float
    if '.' in s:
        val = float(s)
        return int(val) if val.is_integer() else val
    return int(s)
def _tokenize(expr: str) -> List[Token]:
    """
    Tokeniza soportando signo unario +/- . No admite paréntesis.
    Lanza ValueError si encuentra caracteres inválidos.
    """
    expr = expr.strip()
    if any(c in expr for c in '()[{}'):
        raise ValueError("Paréntesis no permitidos en la notación N'mari")
    tokens: List[Token] = []
    i = 0
    n = len(expr)
    def is_op(ch: str) -> bool:
        return ch in '+-*/'
```



<CODEWARS2026/>
LEÓN

```
while i < n:
    ch = expr[i]
    if ch.isspace():
        i += 1
        continue
    # Número (posible signo unario)
    if ch.isdigit() or ch == '.' or (is_op(ch) and (ch in '+-')):
        # Detectar si es signo unario: al inicio o tras otro operador
        if (ch in '+-') and (i == 0 or (tokens and tokens[-1].kind ==
'op') or (not tokens)):
            # signo unario
            j = i + 1
            num_str = ch
            # deben venir dígitos o punto
            if j < n and (expr[j].isdigit() or expr[j] == '.'):
                while j < n and (expr[j].isdigit() or expr[j] == '.'):
                    num_str += expr[j]
                    j += 1
            # validar formato numérico
            if num_str in ['+', '-'] or num_str.count('.') > 1:
                raise ValueError("Número mal formado")
            tokens.append(Token('num', num_str))
            i = j
            continue
        # Si no venían dígitos, entonces no era unario válido
        # caemos a tratarlos como operador normal más adelante
    # número sin signo unario explícito
    if ch.isdigit() or ch == '.':
```

```
j = i
num_str = ''
dot_count = 0
while j < n and (expr[j].isdigit() or expr[j] == '.'):
    if expr[j] == '.':
        dot_count += 1
        if dot_count > 1:
            raise ValueError("Número mal formado")
    num_str += expr[j]
    j += 1
if num_str in ['.', '']:
    raise ValueError("Número mal formado")
tokens.append(Token('num', num_str))
i = j
continue

# Operador binario
if is_op(ch):
    tokens.append(Token('op', ch))
    i += 1
    continue

# Cualquier otro carácter es inválido
raise ValueError(f"Carácter inválido: {ch!r}")

# Validaciones simples de secuencia
if not tokens:
    raise ValueError("Expresión vacía")
if tokens[0].kind == 'op' and tokens[0].value not in ['+', '-']:
    raise ValueError("Expresión no puede iniciar con operador binario")
```



<CODEWARS2026/>
LEÓN

```
if tokens[-1].kind == 'op':
    raise ValueError("Expresión no puede terminar con operador")
# No permitir dos operadores binarios seguidos (salvo que el segundo
se
# hubiese absorbido como signo unario, lo cual ya tratamos arriba)
for a, b in zip(tokens, tokens[1:]):
    if a.kind == 'op' and b.kind == 'op':
        raise ValueError("Operadores consecutivos no permitidos")
return tokens
def _to_rpn(tokens: List[Token]) -> List[Token]:
    """
    Shunting-yard con precedencia estética:
    + y * : precedencia alta (2)
    - y / : precedencia baja (1)
    Asociatividad izquierda.
    """
    prec = {'+': 2, '*': 2, '-': 1, '/': 1}
    out: List[Token] = []
    ops: List[Token] = []
    for t in tokens:
        if t.kind == 'num':
            out.append(t)
        else: # operador
            while ops and ops[-1].kind == 'op' and prec[ops[-1].value] >=
prec[t.value]:
                out.append(ops.pop())
            ops.append(t)
    while ops:
```




<CODEWARS2026/>
LEÓN

```
        out.append(ops.pop())
    return out

def _eval_rpn(rpn: List[Token]) -> Number:
    stack: List[float] = []
    for t in rpn:
        if t.kind == 'num':
            stack.append(float(t.value))
        else:
            if len(stack) < 2:
                raise ValueError("ValueError")
            b = stack.pop()
            a = stack.pop()
            op = t.value
            if op == '+':
                stack.append(a + b)
            elif op == '-':
                stack.append(a - b)
            elif op == '*':
                stack.append(a * b)
            elif op == '/':
                if b == 0:
                    raise ZeroDivisionError("ZeroDivisionError")
                stack.append(a / b)
            else:
                raise ValueError(f"Operador desconocido: {op}")
    if len(stack) != 1:
        raise ValueError("ValueError")
    res = stack[0]
```



<CODEWARS2026/> LEÓN

```
    return int(res) if float(res).is_integer() else res
def evaluate_kardashev(expr: str) -> Number:
    """
    Evalúa 'expr' con la precedencia de la civilización N'mari:
        prioridad alta: +, *
        prioridad baja: -, /
    Todas las operaciones son asociativas por la izquierda.
    """
    tokens = _tokenize(expr)
    rpn = _to_rpn(tokens)
    return _eval_rpn(rpn)
data=input("")
try:
    print(evaluate_kardashev(data))
except ZeroDivisionError:
    print("ZeroDivisionError")
except ValueError:
    print("ValueError")
```

```
C++
#include <bits/stdc++.h>
using namespace std;
struct Token {
    string kind; // "num" o "op"
    string value;
};
double toNumber(const string &s) {
    if (s.find('.') != string::npos) {
        double val = stod(s);
```



<CODEWARS2026/>
LEÓN

```
        return val;
    }
    return stoi(s);
}

vector<Token> tokenize(const string &expr) {
    string e = expr;
    if (e.find('(') != string::npos || e.find(')') != string::npos)
        throw runtime_error("ValueError");
    vector<Token> tokens;
    int i = 0, n = e.size();
    auto isOp = [](char ch) {
        return ch == '+' || ch == '-' || ch == '*' || ch == '/';
    };
    while (i < n) {
        char ch = e[i];
        if (isspace(ch)) { i++; continue; }
        if (isdigit(ch) || ch == '.' || (isOp(ch) && (ch == '+' || ch == '-' || ch == '*')) {
            if ((ch == '+' || ch == '-') && (i == 0 || (!tokens.empty() && tokens.back().kind == "op"))) {
                int j = i + 1;
                string numStr(1, ch);
                if (j < n && (isdigit(e[j]) || e[j] == '.')) {
                    while (j < n && (isdigit(e[j]) || e[j] == '.')) {
                        numStr += e[j++];
                    }
                    tokens.push_back({"num", numStr});
                    i = j;
                }
            }
        }
    }
}
```



<CODEWARS2026/>
LEÓN

```
        continue;
    }
}
if (isdigit(ch) || ch == '.') {
    int j = i;
    string numStr;
    int dotCount = 0;
    while (j < n && (isdigit(e[j]) || e[j] == '.')) {
        if (e[j] == '.') {
            dotCount++;
            if (dotCount > 1) throw
runtime_error("ValueError");
        }
        numStr += e[j++];
    }
    tokens.push_back({"num", numStr});
    i = j;
    continue;
}
if (isOp(ch)) {
    tokens.push_back({"op", string(1, ch)});
    i++;
    continue;
}
throw runtime_error(string("ValueError") + ch);
}
if (tokens.empty()) throw runtime_error("ValueError");
```

<let's code/>





```
        if (tokens.front().kind == "op" && tokens.front().value != "+" &&
tokens.front().value != "-")
            throw runtime_error("ValueError");
        if (tokens.back().kind == "op")
            throw runtime_error("ValueError");
        for (size_t k = 0; k + 1 < tokens.size(); k++) {
            if (tokens[k].kind == "op" && tokens[k + 1].kind == "op")
                throw runtime_error("ValueError");
        }
        return tokens;
    }
    vector<Token> toRPN(const vector<Token> &tokens) {
        map<string, int> prec = {{ "+", 2}, {"*", 2}, {"-", 1}, {"/", 1}};
        vector<Token> out;
        stack<Token> ops;
        for (auto &t : tokens) {
            if (t.kind == "num") out.push_back(t);
            else {
                while (!ops.empty() && prec[ops.top().value] >= prec[t.value])
                {
                    out.push_back(ops.top());
                    ops.pop();
                }
                ops.push(t);
            }
        }
        while (!ops.empty()) {
            out.push_back(ops.top());
        }
    }
}
```



<CODEWARS2026/>
LEÓN

```
        ops.pop();
    }
    return out;
}

double evalRPN(const vector<Token> &rpn) {
    stack<double> st;
    for (auto &t : rpn) {
        if (t.kind == "num") st.push(stod(t.value));
        else {
            if (st.size() < 2) throw runtime_error("ValueError");
            double b = st.top(); st.pop();
            double a = st.top(); st.pop();
            if (t.value == "+") st.push(a + b);
            else if (t.value == "-") st.push(a - b);
            else if (t.value == "*") st.push(a * b);
            else if (t.value == "/") {
                if (b == 0) throw runtime_error("ZeroDivisionError");
                st.push(a / b);
            }
        }
    }
    if (st.size() != 1) throw runtime_error("ValueError");
    return st.top();
}

int main() {
    string expr;
    getline(cin, expr);
    try{
```



<CODEWARS2026/>
LEÓN

```
    auto tokens = tokenize(expr);
    auto rpn = toRPN(tokens);
    double res = evalRPN(rpn);
    if (floor(res) == res) cout << (int)res << "\n";
    else cout << res << "\n";
} catch (const runtime_error &e) {
    cout << e.what() << "\n";
}
return 0;
}
```

Java

```
import java.util.*;
import java.io.*;
class Token {
    String kind; // "num" o "op"
    String value;
    Token(String k, String v) { kind = k; value = v; }
}
public class Un_problema_universal {
    static List<Token> tokenize(String expr) {
        expr = expr.trim();
        if (expr.contains("(") || expr.contains(")"))
            throw new RuntimeException("ValueError");
        List<Token> tokens = new ArrayList<>();
        int i = 0, n = expr.length();
        while (i < n) {
            char ch = expr.charAt(i);
            if (Character.isWhitespace(ch)) { i++; continue; }
```



<CODEWARS2026/>
LEÓN

```
        if (Character.isDigit(ch) || ch == '.' || (isOp(ch) && (ch ==
'+' || ch == '-')))) {
            if ((ch == '+' || ch == '-') && (i == 0 ||
(!tokens.isEmpty() && tokens.get(tokens.size()-1).kind.equals("op")))) {
                int j = i + 1;
                StringBuilder numStr = new StringBuilder();
                numStr.append(ch);
                if (j < n && (Character.isDigit(expr.charAt(j)) ||
expr.charAt(j) == '.')) {
                    while (j < n && (Character.isDigit(expr.charAt(j))
|| expr.charAt(j) == '.')) {
                        numStr.append(expr.charAt(j));
                        j++;
                    }
                    tokens.add(new Token("num", numStr.toString()));
                    i = j;
                    continue;
                }
            }
            if (Character.isDigit(ch) || ch == '.') {
                int j = i;
                StringBuilder numStr = new StringBuilder();
                int dotCount = 0;
                while (j < n && (Character.isDigit(expr.charAt(j)) ||
expr.charAt(j) == '.')) {
                    if (expr.charAt(j) == '.') {
                        dotCount++;
                    }
                }
            }
        }
    }
}
```




<CODEWARS2026/>
LEÓN

```
        if (dotCount > 1) throw new
RuntimeException("ValueError");
    }
    numStr.append(expr.charAt(j));
    j++;
}
tokens.add(new Token("num", numStr.toString()));
i = j;
continue;
}
}
if (isOp(ch)) {
    tokens.add(new Token("op", String.valueOf(ch)));
    i++;
    continue;
}
throw new RuntimeException("ValueError");
}
if (tokens.isEmpty()) throw new RuntimeException("ValueError");
if (tokens.get(0).kind.equals("op") &&
!tokens.get(0).value.equals("+") && !tokens.get(0).value.equals("-"))
    throw new RuntimeException("ValueError");
if (tokens.get(tokens.size()-1).kind.equals("op"))
    throw new RuntimeException("ValueError");
for (int k = 0; k + 1 < tokens.size(); k++) {
    if (tokens.get(k).kind.equals("op") &&
tokens.get(k+1).kind.equals("op"))
        throw new RuntimeException("ValueError");
```



<CODEWARS2026/>
LEÓN

```
    }
    return tokens;
}

static boolean isOp(char ch) {
    return ch == '+' || ch == '-' || ch == '*' || ch == '/';
}

static List<Token> toRPN(List<Token> tokens) {
    Map<String,Integer> prec = new HashMap<>();
    prec.put("+", 2); prec.put("*", 2); prec.put("-", 1);
prec.put("/", 1);
    List<Token> out = new ArrayList<>();
    Stack<Token> ops = new Stack<>();
    for (Token t : tokens) {
        if (t.kind.equals("num")) out.add(t);
        else {
            while (!ops.isEmpty() && prec.get(ops.peek().value) >=
prec.get(t.value)) {
                out.add(ops.pop());
            }
            ops.push(t);
        }
    }
    while (!ops.isEmpty()) out.add(ops.pop());
    return out;
}

static double evalRPN(List<Token> rpn) {
    Stack<Double> st = new Stack<>();
    for (Token t : rpn) {
```

<let's code/>





<CODEWARS2026/>
LEÓN

```
        if (t.kind.equals("num"))
st.push(Double.parseDouble(t.value));
        else {
            if (st.size() < 2) throw new
RuntimeException("ValueError");
            double b = st.pop();
            double a = st.pop();
            switch (t.value) {
                case "+": st.push(a + b); break;
                case "-": st.push(a - b); break;
                case "*": st.push(a * b); break;
                case "/":
                    if (b == 0) throw new
RuntimeException("ZeroDivisionError");
                    st.push(a / b); break;
                default: throw new RuntimeException("ValueError");
            }
        }
    }
    if (st.size() != 1) throw new RuntimeException("ValueError");
    return st.pop();
}

public static void main(String[] args) throws IOException {
    BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
    String expr = br.readLine();
    try {
        List<Token> tokens = tokenize(expr);
```



<CODEWARS2026/>
LEÓN

```
List<Token> rpn = toRPN(tokens);  
double res = evalRPN(rpn);  
if (Math.floor(res) == res) System.out.println((int)res);  
else System.out.println(res);  
} catch (RuntimeException e) {  
    System.out.println(e.getMessage());  
}  
}  
}
```

<let's code/>



20

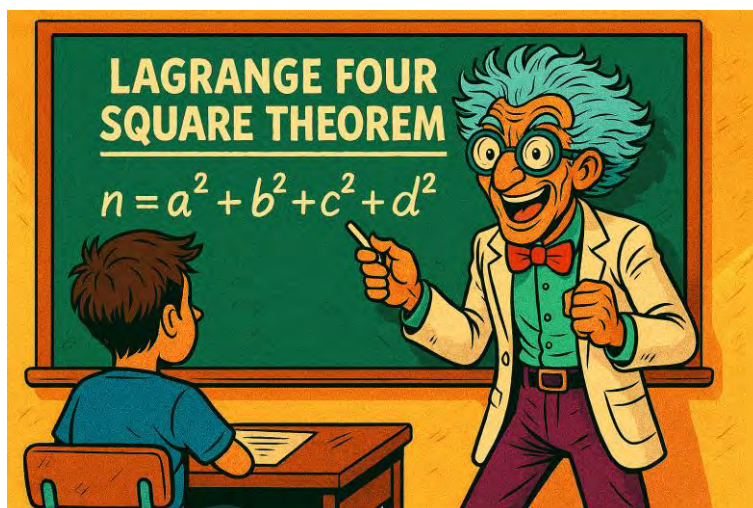
Lagrange's mouthshutter

12 points

Introduction

Recently, you had an argument with your mathematics teacher. He not only demands knowledge of all the theorems, but he has also become a devoted constructivist!

After you recited the Lagrange Four Square Theorem by heart, he now demands a computer program to obtain such a decomposition, to prove that you really understand the topic. How frustrating!



You remember the theorem well: Any positive integer can be decomposed as the sum of four squares of integers.

For example:

$$23 = 3^2 + 3^2 + 2^2 + 1^2$$

Or

$$1 = 1^2 + 0^2 + 0^2 + 0^2$$

In some cases, there are different alternatives:

$$59 = 7^2 + 3^2 + 1^2 + 0^2$$

$$59 = 5^2 + 5^2 + 3^2 + 0^2$$

$$59 = 5^2 + 4^2 + 3^2 + 3^2$$

That is:

$$N = a^2 + b^2 + c^2 + d^2$$

Exercise

To shut the teacher's mouth up, your task is to generate a program that returns the decomposition as 4 squares of a number (up to 1,000,000) passed as input, choosing the decomposition where the highest number is the largest first square. If a number has several alternatives with the same highest square, the option with the second highest square should be chosen and so on. For example, having as input the number 317 the ordered decompositions are:

$$317 = 16^2 + 6^2 + 5^2 + 0^2$$

$$317 = 16^2 + 6^2 + 4^2 + 3^2$$

$$317 = 14^2 + 11^2 + 0^2 + 0^2$$

...

Thus, the only correct answer is the first one because 16 is the highest first squared integer and, from those which have this same highest first squared integer, as they also have the same second squared integer, comparing the next value, 5 is the highest third squared integer.

A single solution will always be printed.

Input

A single number from 1 to 999999.

Output

The lowest value decomposition of the input number in squared integers following the Lagrange's Theorem.

As indicated, in some cases, there are different alternatives:

```
59 = 72 + 32 + 12 + 02  
59 = 52 + 52 + 32 + 02  
59 = 52 + 42 + 32 + 32  
...
```

When this happens, the appropriate one is that with the lowest "a", then the lowest "b", etc. Therefore a must be $\geq b$ and b must be $\geq c$ and c must be $\geq d$:

$$N = a^2 + b^2 + c^2 + d^2$$

Therefore in the previous example, the expected output is

$$59 = 5^2 + 4^2 + 3^2 + 3^2$$

Example 1

Input

```
1
```

Output

$$1 = 1^2 + 0^2 + 0^2 + 0^2$$

Example 2

Input

```
59
```

Output

$$59 = 5^2 + 4^2 + 3^2 + 3^2$$

Example 3

Input

23

Output

23 = 3^2 + 3^2 + 2^2 + 1^2

Solutions

Python

```
import math

def descomposition_lagrange_minmax(n: int):
    """
    Devuelve la descomposición  $n = a^2 + b^2 + c^2 + d^2$ 
    cumpliendo  $a \geq b \geq c \geq d \geq 0$ 
    y donde la tupla (a,b,c,d) es la lexicográficamente mínima.
    """
    # El mayor cuadrado ( $a^2$ ) debe cumplir  $a^2 \leq n$ .
    # Además, para 4 cuadrados, el valor mínimo posible de a es
    ceil(sqrt(n/4)).
    a_min = math.isqrt((n + 3) // 4)
    # Recorremos a desde su menor valor posible
    for a in range(a_min, math.isqrt(n) + 1):
        aa = a * a
        #  $b \leq a$ 
        b_max = min(a, math.isqrt(n - aa))
        # b de 0 ... b_max (minimiza b)
        for b in range(0, b_max + 1):
            bb = aa + b * b
```




<CODEWARS2026/>
LEÓN

```
        if bb > n:
            break
        #  $c \leq b$ 
        c_max = min(b, math.isqrt(n - bb))
        # c de 0 ... c_max (minimiza c)
        for c in range(0, c_max + 1):
            cc = bb + c * c
            if cc > n:
                break
            rest = n - cc
            d = math.isqrt(rest)
            #  $d \leq c$  y cuadrado perfecto
            if d <= c and d*d == rest:
                return (a, b, c, d)

    return None

# Uso
number = int(input().strip())
if 1 <= number <= 999999:
    sol = descomposition_lagrange_minmax(number)
    if sol:
        a,b,c,d = sol
        print(f"{number} = {a}^2 + {b}^2 + {c}^2 + {d}^2")
else:
    print("The number must be between 1 and 999999.")
```

C++

```
#include <bits/stdc++.h>
using namespace std;
// Raíz cuadrada entera segura (evita imprecisiones de casting)
```

<let's code/>





<CODEWARS2026/>
LEÓN

```
static inline int isqrt_ll(long long x) {
    // sqrtl con long double es robusto y rápido para nuestro rango
    return static_cast<int>(sqrtl(static_cast<long double>(x)));
}

static inline int ceil_isqrt_div4(int n) {
    // ceil(sqrt(n/4)) -> la cota inferior del mayor sumando
    // equivalente entero: ceil( sqrt(n/4) ) = ceil( sqrt(n) / 2 )
    // Usamos una forma entera estable:
    // a_min = smallest a s.t. 4*a*a >= n
    int a = isqrt_ll((n + 3) / 4);          // aproximación por entero
    // Ajuste fino por si hiciera falta
    while (1LL * (a - 1) * (a - 1) * 4 >= n && a > 0) --a;
    while (1LL * a * a * 4 < n) ++a;
    return a;
}

optional<array<int,4>> lagrange_decomposition_minlex(int n) {
    // a es el mayor sumando; para minimizar (a,b,c,d) con a>=b>=c>=d,
    // empezamos en el menor a posible (ceil(sqrt(n/4))) y vamos subiendo.
    const int a_min = ceil_isqrt_div4(n);
    const int a_max = isqrt_ll(n);
    for (int a = a_min; a <= a_max; ++a) {
        long long aa = 1LL * a * a;
        // b <= a y b^2 <= n - a^2
        int b_upper = min(a, isqrt_ll(n - aa));
        for (int b = 0; b <= b_upper; ++b) {           // b crece ->
            minimiza b
            long long bb = aa + 1LL * b * b;
            if (bb > n) break;
        }
    }
}
```



<CODEWARS2026/>
LEÓN

```
// c <= b y c^2 <= n - a^2 - b^2
int c_upper = min(b, isqrt_ll(n - bb));
for (int c = 0; c <= c_upper; ++c) {           // c crece ->
minimiza c
    long long cc = bb + 1LL * c * c;
    if (cc > n) break;
    long long rest = n - cc;
    int d = isqrt_ll(rest);
    // exigir orden a>=b>=c>=d y que d^2 == rest
    if (d <= c && 1LL * d * d == rest) {
        return array<int,4>{a, b, c, d};      // primera
encontrada = lexicográficamente mínima
    }
}
}
}
return nullopt;
}
int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    long long number;
    if (!(cin >> number)) return 0;
    if (1 <= number && number <= 999'999) {
        auto sol =
lagrange_decomposition_minlex(static_cast<int>(number));
        if (sol.has_value()) {
            auto v = *sol; // v[0]=a >= v[1]=b >= v[2]=c >= v[3]=d
```



<CODEWARS2026/>
LEÓN

```
        cout << number << " = "  
            << v[0] << "^2 + "  
            << v[1] << "^2 + "  
            << v[2] << "^2 + "  
            << v[3] << "^2\n";  
    }  
    } else {  
        cout << "The number must be between 1 and 999999.\n";  
    }  
    return 0;  
}
```

Java

```
import java.io.BufferedReader;  
import java.io.IOException;  
import java.io.InputStreamReader;  
import java.util.Arrays;  
public class lagrange {  
    // Raíz entera segura (evita errores de redondeo al castear)  
    private static int isqrt(long x) {  
        return (int) Math.sqrt((double) x);  
    }  
    private static int[] lagrangeDecomposition(int n) {  
        final int A = isqrt(n);  
        for (int a = 0; a <= A; a++) {  
            long aa = 1L * a * a;  
            for (int b = 0; b <= a; b++) { // b <= a  
                long bb = aa + 1L * b * b;  
                if (bb > n) break;  
            }  
        }  
    }  
}
```




<CODEWARS2026/>
LEÓN

```
        for (int c = 0; c <= b; c++) {                // c <= b
            long cc = bb + 1L * c * c;
            if (cc > n) break;
            long rest = n - cc;
            int d = isqrt(rest);
            // d <= c + exactitud
            if (d <= c && 1L * d * d == rest) {
                return new int[]{a, b, c, d};          // ya en orden
            }
        }
    }
}
return null;
}

public static void main(String[] args) throws IOException {
    BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
    String line = br.readLine();
    if (line == null) return;
    long number = Long.parseLong(line.trim());
    if (1 <= number && number <= 999_999) {
        int[] res = lagrangeDecomposition((int) number);
        if (res != null) {
            // Orden descendente, como haces con sorted(reverse=True)
            Arrays.sort(res);
            // Arrays.sort ascendente: invertimos al imprimir
            System.out.printf("%d = %d^2 + %d^2 + %d^2 + %d^2%n",
```



<CODEWARS2026/>
LEÓN

```
        number, res[3], res[2], res[1], res[0]);  
    }  
    } else {  
        System.out.println("The number must be between 1 and  
999999.");  
    }  
}  
}
```

<let's code/>



21

Supreme numbers

12 points

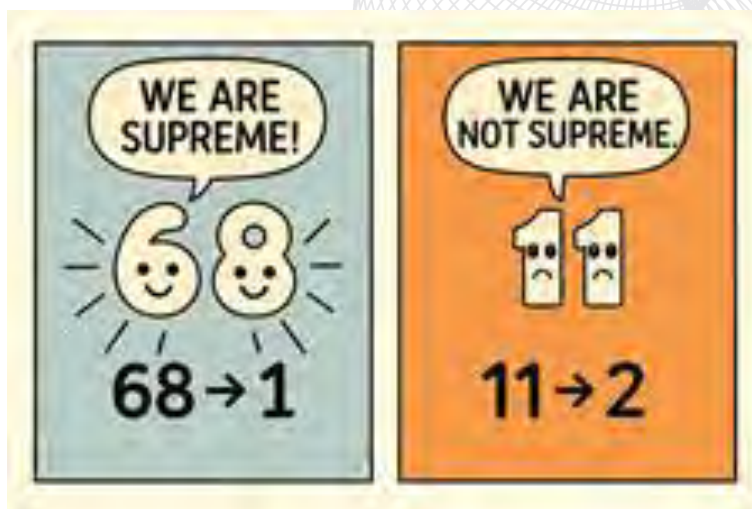
Introduction

Numeroland is the country of numbers. Numbers are generally happy, however inside numeroland inhabitants fall in different categories: There are supreme class numbers and there are normal class numbers.

Unfortunately not all numbers are supreme. For those of you who don't know, a supreme number is a number in which, if we repeatedly add the sum of the squares of its digits, at the end we get one (1). For example:

- 68 is a supreme number. why? Because $6^2 + 8^2 = 36 + 64 = 100$, and then $1^2 + 0^2 + 0^2 + 0^2 = 1$.
- 28 is a supreme number, because $2^2 + 8^2 = 4 + 64 = 68$. And then 68, as we have seen, is happy and in the end it adds up to 1.
- The 23 is a supreme number, because $2^2 + 3^2 = 4 + 9 = 13$. And then 13, is $1^2 + 3^2 = 10$ $1^2 + 0^2$ is 1.
- The number 11, on the other hand, is not supreme because $1^2 + 1^2 = 2$.

There are numbers that are supreme by birth: they alone are supreme (1, 28, 68, 100...). If they are grouped, they could stop being so (for instance $68 + 1 = 69$, as 69 is not supreme, both 68 and 1 stop being supreme. On the other hand non supreme number may become supreme when joining. Note that supreme numbers by birth are very selfing: they will not help to construct new supreme numbers. For instance with numbers 900 and 100 in the input, there will not be a chance to construct a new supreme number, because 100 is supreme by itself.



Exercise

Since we want all possible numbers to be supreme, you are asked to write a program that, given a series of positive numbers, obtain the groups of numbers that added together get as many supreme numbers as possible.

Input

The program will receive a list of numbers, finished by number "-1".

For example an input like:

```
34
50
50
-1
```

Note: The last -1 is to indicate where the list ends, it should not be taken into consideration. In this case all three numbers are initially not supreme.

Output

With the previous input, the program will indicate the numbers that are already 'supreme' and which unions of numbers make a new number supreme. The structure of the output will be:

```
Supreme numbers:  
<list of numbers already supreme>  
Unions that make supreme numbers:  
<list of griuping that make supreme numbers>
```

Therefore for this example the output would be:

```
Supreme numbers:  
None  
Unions that make supreme numbers:  
50+50
```

Note: The output should be like this, separated by the symbol "+".

Example 1

Input

```
100  
34  
50  
-1
```

Output

```
Supreme numbers:  
100  
Unions that make supreme numbers:  
None
```

Example 2

Input

```
100  
100  
50
```

```
50  
-1
```

Output

```
Supreme numbers:  
100, 100  
Unions that make supreme numbers:  
50+50
```

Example 3

Input

```
5  
5  
50  
50  
-1
```

Output

```
Supreme numbers:  
None  
Unions that make supreme numbers:  
5+5  
50+50
```

Solutions

Python

```
import itertools  
  
def es_feliz(num):  
    """Verifica si un número es feliz (supreme)."""  
    vistos = set()  
    while num != 1 and num not in vistos:  
        vistos.add(num)  
        num = sum(int(digit)**2 for digit in str(num))
```



<CODEWARS2026/> LEÓN

```
    return num == 1
def agrupar_supremos(numeros):
    """Agrupa los números en supremos, verificando combinaciones
    posibles."""
    supremos = []
    no_felices = []
    # Clasificar los números en felices y no felices
    for num in numeros:
        if es_feliz(num):
            supremos.append(num)
        else:
            no_felices.append(num)
    # Intentar encontrar combinaciones de números no felices que formen un
    número supreme
    uniones = []
    # Probar todas las combinaciones posibles de números no felices
    for r in range(2, len(no_felices) + 1): # Combinaciones de tamaño 2
        hasta el tamaño de la lista
        for combinacion in itertools.combinations(no_felices, r):
            # print(combinacion)
            suma = sum(combinacion)
            if es_feliz(suma):
                # Solo agregamos la combinación si el resultado no está ya
                en supremos
            # print(supremos)
            # if suma not in supremos:
            uniones.append(f"{'+'.join(map(str, combinacion))}")
    print(uniones)
```



<CODEWARS2026/>
LEÓN

```
#                 supremos.append(suma)
# Eliminar los números de la lista de no felices para
no volver a usarlos
        for num in combinacion:
            no_felices.remove(num)

print(uniones)
return supremos, uniones
def main():
    # Leemos la entrada
    numeros = []
    while True:
        num = int(input())
        if num == -1:
            break
        numeros.append(num)
    supremos, uniones = agrupar_supremos(numeros)
    # Mostrar los resultados
    print("Supreme numbers:")
    if supremos:
        print(", ".join(map(str, supremos)))
    else:
        print("None")
    print("Unions that make supreme numbers:")
    if uniones:
        for union in uniones:
            print(union)
    else:
        print("None")
```




<CODEWARS2026/> LEÓN

```
if __name__ == "__main__":  
    main()
```

C++

```
#include <iostream>  
#include <vector>  
#include <set>  
#include <sstream>  
#include <iterator>  
#include <algorithm>  
  
// Función para verificar si un número es feliz (supreme)  
bool esFeliz(int num) {  
    std::set<int> vistos;  
    while (num != 1 && vistos.find(num) == vistos.end()) {  
        vistos.insert(num);  
        int suma = 0;  
        while (num > 0) {  
            int digito = num % 10;  
            suma += digito * digito;  
            num /= 10;  
        }  
        num = suma;  
    }  
    return num == 1;  
}  
  
// Función para encontrar las uniones de números "supremes"  
void agruparSupremos(const std::vector<int>& numeros) {  
    std::vector<int> supremos;  
    std::vector<int> noFelices;
```



<CODEWARS2026/>
LEÓN

```
std::vector<std::string> uniones;
// Clasificar los números en felices (supremos) y no felices
for (int num : numeros) {
    if (esFeliz(num)) {
        supremos.push_back(num);
    } else {
        noFelices.push_back(num);
    }
}

// Intentar encontrar combinaciones de números no felices que sumen un
número "supreme"
for (size_t r = 2; r <= noFelices.size(); ++r) {
    for (size_t i = 0; i <= noFelices.size() - r; ++i) {
        int suma = 0;
        std::vector<int> combinacion;
        for (size_t j = 0; j < r; ++j) {
            combinacion.push_back(noFelices[i + j]);
            suma += noFelices[i + j];
        }
        if (esFeliz(suma)) {
            std::ostringstream oss;
            for (size_t j = 0; j < combinacion.size(); ++j) {
                oss << combinacion[j];
                if (j < combinacion.size() - 1) {
                    oss << "+";
                }
            }
            uniones.push_back(oss.str());
        }
    }
}
```



<CODEWARS2026/>
LEÓN

```
    }
}
}
// Mostrar los resultados
if (!supremos.empty()) {
    std::cout << "Supreme numbers:\n";
    for (size_t i = 0; i < supremos.size(); ++i) {
        std::cout << supremos[i];
        if (i < supremos.size() - 1) {
            std::cout << ", ";
        }
    }
    std::cout << std::endl;
} else {
    std::cout << "Supreme numbers:\nNone\n";
}
if (!uniones.empty()) {
    std::cout << "Unions that make supreme numbers:\n";
    for (const std::string& unionStr : uniones) {
        std::cout << unionStr << std::endl;
    }
} else {
    std::cout << "Unions that make supreme numbers:\nNone\n";
}
}
int main() {
    std::vector<int> numeros;
    int num;
```



<CODEWARS2026/>
LEÓN

```
// Leer la entrada
while (std::cin >> num && num != -1) {
    numeros.push_back(num);
}
// Llamar a la función para agrupar los números supremes y las uniones
agruparSupremos(numeros);
return 0;
}
```

Java

```
import java.util.*;
import java.util.stream.Collectors;
public class NumeroSupreme {
    /** Verifica si un número es feliz (supreme). */
    public static boolean esFeliz(long num) {
        Set<Long> vistos = new HashSet<>();
        while (num != 1 && !vistos.contains(num)) {
            vistos.add(num);
            num = sumaCuadradosDigitos(num);
        }
        return num == 1;
    }
    private static long sumaCuadradosDigitos(long num) {
        long suma = 0;
        if (num < 0) num = -num; // por si llega algún negativo distinto
        while (num > 0) {
            long d = num % 10;
            suma += d * d;
        }
    }
}
```

<let's code/>




```
        num /= 10;
    }
    return suma;
}

/** Contenedor de resultados: lista de supremos y lista de uniones. */
public static class Resultado {
    public final List<Long> supremos;
    public final List<String> uniones;
    public Resultado(List<Long> supremos, List<String> uniones) {
        this.supremos = supremos;
        this.uniones = uniones;
    }
}

/**
 * Agrupa los números en supremos, verificando combinaciones posibles
de no felices.
 * Replica la semántica del código Python:
 * - Clasifica en felices (supremos) y no felices.
 * - Prueba combinaciones de tamaño 2..N de los no felices (snapshot
por cada r).
 * - Si la suma de una combinación es feliz, se añade la unión y se
eliminan
 * dichos elementos de la lista de no felices para evitar
reutilización posterior.
 */
public static Resultado agruparSupremos(List<Long> numeros) {
    List<Long> supremos = new ArrayList<>();
    List<Long> noFelices = new ArrayList<>();
```

```
// Clasificar los números en felices y no felices
for (Long num : numeros) {
    if (esFeliz(num)) {
        supremos.add(num);
    } else {
        noFelices.add(num);
    }
}

// Intentar encontrar combinaciones de números no felices que
formen un número supreme
List<String> uniones = new ArrayList<>();
// Probar todas las combinaciones posibles de números no felices
(tamaño 2..len)
for (int r = 2; r <= noFelices.size(); r++) {
    // Generamos combinaciones a partir de un "snapshot" para
    replicar el comportamiento del Python
    List<List<Long>> combos = generarCombinaciones(new
    ArrayList<>(noFelices), r);
    for (List<Long> combinacion : combos) {
        long suma =
        combinacion.stream().mapToLong(Long::longValue).sum();
        if (esFeliz(suma)) {
            // Añadir unión en formato "a+b+..."
            String union = combinacion.stream()
                .map(String::valueOf)

.collect(Collectors.joining("+"));
            uniones.add(union);
        }
    }
}
```

```
        // Eliminar números usados de noFelices (una
ocurrencia por valor)
        for (Long val : combinacion) {
            noFelices.remove(val); // elimina la primera
coincidencia
        }
    }
}

return new Resultado(supremos, uniones);
}
/**
 * Genera todas las combinaciones de tamaño r a partir de la lista
base.
 * Implementación recursiva sin librerías externas.
 */
private static List<List<Long>> generarCombinaciones(List<Long> base,
int r) {
    List<List<Long>> resultado = new ArrayList<>();
    backtrackCombinaciones(base, r, 0, new ArrayList<>(), resultado);
    return resultado;
}

private static void backtrackCombinaciones(List<Long> base, int r, int
inicio,
                                List<Long> actual,
List<List<Long>> resultado) {
    if (actual.size() == r) {
        resultado.add(new ArrayList<>(actual));
    }
}
```

```
        return;
    }
    for (int i = inicio; i < base.size(); i++) {
        actual.add(base.get(i));
        backtrackCombinaciones(base, r, i + 1, actual, resultado);
        actual.remove(actual.size() - 1);
    }
}

/** Flujo principal: entrada por stdin, salida en español. */
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    List<Long> numeros = new ArrayList<>();
    // Lectura hasta -1
    while (true) {
        if (!sc.hasNextLong()) break; // fin de entrada
        long num = sc.nextLong();
        if (num == -1) break;
        numeros.add(num);
    }
    Resultado res = agruparSupremos(numeros);
    // Mostrar resultados
    System.out.println("Supreme numbers:");
    if (!res.supremos.isEmpty()) {
        System.out.println(res.supremos.stream()
                                .map(String::valueOf)
                                .collect(Collectors.joining(",
"")));
    } else {
```




<CODEWARS2026/>
LEÓN

```
        System.out.println("None");
    }
    System.out.println("Unions that make supreme numbers:");
    if (!res.uniones.isEmpty()) {
        for (String union : res.uniones) {
            System.out.println(union);
        }
    } else {
        System.out.println("None");
    }
}
}
```

<let's code/>



22

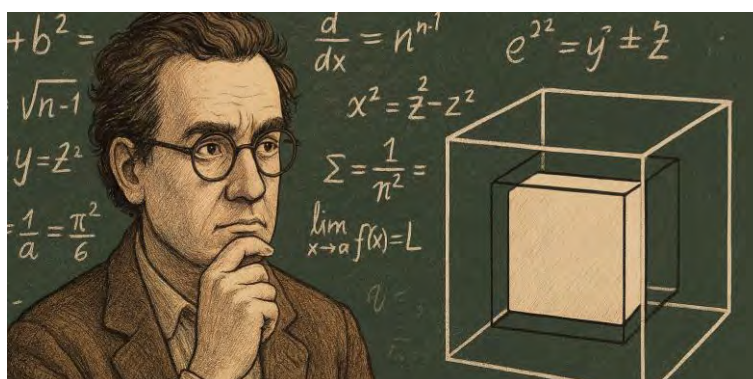
The Digits of the Perfect Square and Cube

12 points

Introduction

A perfect square is an integer that is the result of squaring another integer. That is, a number 'n' is a perfect square if there exists an integer k such that $n=k^2$. '9', '16' and '25' are perfect squares.

A perfect cube is an integer that is the result of raising another integer to the third power. That is, a number 'n' is a perfect cube if there exists an integer k such that $n=k^3$. '8', '27' and '64' are perfect cubes.



Exercise

You are asked to write a program that finds the smallest number that is both a perfect square and a perfect cube and that has (at least once) the digits indicated as input parameters. For example, if the numbers 1, 2, 4, and 6 are specified, you must find a number that is a perfect square and a perfect cube and has the digits 1, 2, 4, and 6 at least once.

The program will check up to the number 99999999. If no number is found, the program will write:

Could not find the specified Cube&Square number

Input

The program will receive an input like:

```
1;2;4;6
```

Note that inputted values are digits between 0 and 9, separated by ';'.

Output

...the output will be something like

<number>;<square root of number>;<cubic root of number>

For example:

```
262144;512;64
```

Note: This is because 262144 is 64^3 and at the same time 262144 is 512^2 . And at the same time 262144 is written with 1, 2, 4 and 6. Take into account, that you could have other digits different than 1,2,4 and 6, but AT LEAST these four need to be present once.

Example 1

Input

```
1;2;4;6
```

Output

```
262144;512;64
```

Example 2

Input

1;2;3

Output

11390625;3375;225

Example 3

Input

1

Output

1;1;1

Solutions

Python

```
#!/usr/bin/python3
def all_digits_in(digits, number):
    number_str = str(number)
    for digit in digits:
        if digit not in number_str:
            return False
    return True
def perfect_root(number, power):
    return pow(round(pow(number, 1/power)), power) == number
def test():
    print(perfect_root(262144, 2))
```




<CODEWARS2026/>
LEÓN

```
print(perfect_root(262144, 3))
print(all_digits_in(['1', '2', '4', '6'], 262144))
def main():
    digits = [i for i in input().split(';')]
    test_number = 0
    while True:
        if perfect_root(test_number, 3):
            if perfect_root(test_number, 2):
                if all_digits_in(digits, test_number):
                    print(str(test_number)+";"+str(round(pow(test_number,
1/2))))+";"+str(round(pow(test_number, 1/3))))
                    return
                test_number += 1
if __name__ == '__main__':
    main()
    #test()
```

C++

```
#include <iostream>
#include <cmath>
#include <vector>
#include <sstream>
bool allDigitsIn(const std::vector<std::string>& digits, long long number)
{
    std::string numberStr = std::to_string(number);
    for (const auto& digit : digits) {
        if (numberStr.find(digit) == std::string::npos) {
            return false;
        }
    }
}
```

<let's code/>





<CODEWARS2026/>
LEÓN

```
    }
    return true;
}

bool perfectRoot(long long number, int power) {
    long long root = std::llround(std::pow(number, 1.0 / power));
    return std::pow(root, power) == number;
}

int main() {
    std::string line;
    std::getline(std::cin, line);
    std::vector<std::string> digits;
    std::stringstream ss(line);
    std::string item;
    while (std::getline(ss, item, ';')) {
        digits.push_back(item);
    }
    long long testNumber = 0;
    while (true) {
        if (perfectRoot(testNumber, 3)) {
            if (perfectRoot(testNumber, 2)) {
                if (allDigitsIn(digits, testNumber)) {
                    long long sqrt = std::llround(std::sqrt(testNumber));
                    long long cbrt = std::llround(std::pow(testNumber, 1.0
/ 3));

                    std::cout << testNumber << ";" << sqrt << ";" << cbrt
<< std::endl;

                    break;
                }
            }
        }
    }
}
```

<let's code/>





<CODEWARS2026/>
LEÓN

```
        }  
    }  
    testNumber++;  
}  
return 0;  
}
```

Java

```
import java.util.Scanner;  
  
public class The_Digits_of_the_Perfect_Square_and_Cube {  
    public static boolean allDigitsIn(String[] digits, long number) {  
        String numberStr = Long.toString(number);  
        for (String digit : digits) {  
            if (!numberStr.contains(digit)) {  
                return false;  
            }  
        }  
        return true;  
    }  
  
    public static boolean perfectRoot(long number, int power) {  
        long root = Math.round(Math.pow(number, 1.0 / power));  
        return Math.pow(root, power) == number;  
    }  
  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        String[] digits = sc.nextLine().split(";");  
        long testNumber = 0;  
        while (true) {  
            if (perfectRoot(testNumber, 3)) {
```

<let's code/>





<CODEWARS2026/>
LEÓN

```
        if (perfectRoot(testNumber, 2)) {  
            if (allDigitsIn(digits, testNumber)) {  
                long sqrt = Math.round(Math.pow(testNumber, 0.5));  
                long cbrt = Math.round(Math.pow(testNumber, 1.0 /  
3));  
  
                System.out.println(testNumber + ";" + sqrt + ";" +  
cbrt);  
  
                break;  
            }  
        }  
        testNumber++;  
    }  
    sc.close();  
}  
}
```

<let's code/>



24

The Enigmatic multiplication

14 points

Introduction

Sherlock Holmes is investigating an strange case and has found many papers with Enigmatic Multiplications in them. He soon discovers that in them the numbers have been replaced by letters. He has called you as he knows you are one of those rare programmers that exist in the 23IX century...



Exercise

You are asked to develop a program capable of discovering a multiplication in which the numbers have been replaced by letters. It will be a multiplication like the following:

```

  1 2
x 3 4
---
 4 8
3 6
----
4 0 8

```

but with letters:

```

A B

```

```
C D
---
D G
C H
-----
D F G
```

Your program will receive a two-operand multiplication and return the 'lowest' solution.

Input

The program will receive as input a series of lines corresponding to the multiplicands, the partial results, and the final result. The input will end with a semicolon (";"). The digits will be separated among them with an space.

To explain the location of each digit, in this example the left spaces will be filled with dots. For instance the previous multiplication's input will be inputted like:

```
. A B
. C D
. D G
C H
D F G
;
```

Note 1: This exercise will ALWAYS receive a 2 number multiplication (but each number could have 2, 3 or more 2 digits, note it), therefore the two first lines are the numbers that will be multiplied. The number of intermediate lines depend on the number of digits of the numbers multiplied.

Note 2: Separator lines ("---") are not introduced. They appear in examples only for clarity.

Output

The program will show the output with the following format:

```
A=1;B=2;C=3;D=4;F=0;G=8;H=6
```

It should be ordered by letter

Note 1: The multiplicands will NOT start by 0

Note 2: To avoid that an input may generate different solutions, you are asked to show only the 'lowest' solution. That is, the solution where the A is lowest, B is lowest, C is lowest... For instance if solutions may be

```
A=1;B=2;C=1  
A=1;B=3;C=0
```

You will show the first line: That with A being lowest, in case of equality, that with B being lowest, etc.

Example 1

Input

```
. A B  
. C D  
. D G  
C H  
D F G  
;
```

Output

```
A=1;B=2;C=3;D=4;F=0;G=8;H=6
```

Example 2

Input

```
. . A B C  
. . . D A  
. E E F G  
A F A H  
A C G H G  
;
```

Output

A=7;B=8;C=6;D=9;E=1;F=0;G=2;H=4

Solutions

Python

```
import itertools

def count_leading_dots(tokens):
    """Cuenta los puntos ('.') a la izquierda hasta el primer token no
    punto."""
    c = 0
    for t in tokens:
        if t == '.':
            c += 1
        else:
            break
    return c

def parse_input():
    lines = []
    while True:
        line = input().rstrip()
        if line == ";":
            break
        if line: # ignorar líneas vacías
            lines.append(line)
    return lines

def extract_letters(lines):
    letters = set()
    for line in lines:
        for ch in line:
```



```
        if ch.isalpha():
            letters.add(ch)
    return sorted(letters)
def tokens_from_line(line):
    """Devuelve la lista de tokens separados por espacios (cada token será
    '.' o una letra)."""
    return line.split()
def build_number_from_tokens(tokens, mapping):
    """Construye la cadena de dígitos (sin puntos) a partir de tokens
    usando mapping."""
    s = []
    for t in tokens:
        if t == '.':
            continue
        # t debe ser una letra (según la especificación)
        if t.isalpha():
            s.append(mapping[t])
        else:
            # por seguridad, si hubiera dígito directo
            s.append(t)
    return ''.join(s)
def find_leading_letters(lines):
    """
    Devuelve el conjunto de letras que son la primera (más a la izquierda)
    cifra de algún número que contenga más de una cifra.
    """
    leading = set()
    for line in lines:
```



<CODEWARS2026/>
LEÓN

```
tokens = tokens_from_line(line)
# contar tokens relevantes (no '.')
relevant = [t for t in tokens if t != '.']
if len(relevant) <= 1:
    continue
# buscar la primera token relevante; si es letra, anotarla
for t in tokens:
    if t == '.':
        continue
    if t.isalpha():
        leading.add(t)
    # si es dígito literal (no esperado), no añadimos
    break
return leading
def solve(lines):
    """
    En letters se saca el listado de letras ordenado, A,B,C,D... sin
    ningun orden
    """
    letters = extract_letters(lines)
    if len(letters) > 10:
        return None # imposible: más de 10 letras
    # tokens por línea para reutilizar
    """
    Genera una lista de listas como: [['.', 'A', 'B'], [['.', 'C', 'D'],
    ...]
    """
    lines_tokens = [tokens_from_line(line) for line in lines]
```



<CODEWARS2026/>
LEÓN

```
# conjunto de letras que no pueden valer 0 (leading letters of multi-
digit numbers)
leading_letters = find_leading_letters(lines)
# identificamos multiplicando(s) y resultado final:
# según el enunciado las dos primeras líneas son los factores y la
última es el resultado final
# (las demás son parciales). Nos ceñimos a verificar multiplicand1 *
multiplicand2 == final_result.
# Si deseas comprobar parciales, podemos añadirlo.
# Convertiremos cada línea en número (ignorando líneas vacías).
for perm in itertools.permutations('0123456789', len(letters)):
    mapping = dict(zip(letters, perm))
    # poda: si alguna letra líder se mapea a '0', saltar
    if any(mapping[l] == '0' for l in leading_letters):
        continue
    try:
        numbers = []
        for tokens in lines_tokens:
            #tokens es ['.', 'A', 'B'] y similar y numstr es un
            # posible número que podría encajar con .AB
            numstr = build_number_from_tokens(tokens, mapping)
            if numstr == '':
                # línea sin números (posible), la ignoramos
                continue
            # evitar números con ceros a la izquierda (si la
            # representación tiene >1 dígito y empieza con '0')
            if len(numstr) > 1 and numstr[0] == '0':
                raise ValueError("leading zero")
```

```
        numbers.append(int(numstr))
    if len(numbers) < 3:
        continue
    multiplicand1 = numbers[0]
    multiplicand2 = numbers[1]
    final_result = numbers[-1]
    if multiplicand1 * multiplicand2 != final_result:
        continue
    #print(mapping,"--",tokens,"--",numstr)
    # ===== Verificación de productos parciales con alineado por
puntos =====
    # Asumimos:
    #   - total_columns = número de tokens de la línea del
resultado (todas deberían tener el mismo ancho).
    #   - los parciales son lines_tokens[2:-1], en orden: primero
el parcial de las unidades.
    total_columns = len(lines_tokens[-1])
    partial_lines = lines_tokens[2:-1]
    # reconstruimos la cadena del segundo factor para inspeccionar
sus dígitos de derecha a izquierda
    m2_str = build_number_from_tokens(lines_tokens[1], mapping)
    if m2_str == '':
        # segundo factor vacío → inválido
        continue
    # Si hay líneas parciales presentadas, deben corresponder a
los dígitos de m2.
    # Permitimos que haya el mismo número de parciales que dígitos
de m2.
```



```
# (Si tu formato permite omitir parciales que salen 0, este
bloque ya lo contempla más abajo.)
if len(partial_lines) != len(m2_str):
    # Si quieres permitir discrepancias, puedes relajar esta
    restricción,
    # pero por defecto exigimos mismo número de parciales que
    dígitos en multiplicand2.
    continue
# Verificamos cada parcial i (0 = unidades)
ok_partials = True
for i, partial_tokens in enumerate(partial_lines):
    digit = int(m2_str[-1 - i]) # dígito i desde la derecha
    expected_value_str = str(multiplicand1 * digit)
    # expected_left_dots = columnas totales - (longitud del
    valor sin shift + shift i)
    expected_left_dots = total_columns -
    (len(expected_value_str) + i)
    # conteo real de padding
    actual_left_dots = count_leading_dots(partial_tokens)
    actual_numstr = build_number_from_tokens(partial_tokens,
mapping)

    # Validaciones:
    # 1) padding a la izquierda debe coincidir
    # - salvo el caso especial digit==0 con línea vacía:
    permitimos que actual_numstr == ''
    # y entonces solo exigimos que la línea sean todo
    puntos (padding completo).
    if digit == 0:
```

```
# Aceptamos:
#   a) línea completamente vacía (solo puntos) ->
actual_numstr == ''
#   b) línea con "0" alineado exactamente ->
actual_numstr == '0' y padding coincide
only_dots = all(t == '.' for t in partial_tokens)
if only_dots:
    # Para "solo puntos", aceptamos como válido
    (representa 0 desplazado)
    continue
else:
    # Debe ser un "0" exactamente y padding exacto
    if not (actual_numstr == '0' and actual_left_dots
== expected_left_dots):
        ok_partials = False
        break
else:
    # Para digit != 0 debe coincidir valor y padding
    if not (actual_numstr == expected_value_str and
actual_left_dots == expected_left_dots):
        ok_partials = False
        break
if not ok_partials:
    continue
# ===== Fin verificación de productos parciales =====
return mapping
except ValueError:
    continue
```



<CODEWARS2026/> LEÓN

```
        return None

def main():
    lines = parse_input()
    mapping = solve(lines)
    if mapping:
        # imprimir en orden alfabético sin espacio final: A=6;B=3;...
        out = ';'.join(f"{k}={mapping[k]}" for k in sorted(mapping))
        print(out)
    else:
        print("No solution found")

if __name__ == "__main__":
    main()
```

C++

```
#include <bits/stdc++.h>
using namespace std;
// === Utilidades de tokenización y construcción ===
static vector<string> tokens_from_line(const string& line) {
    vector<string> tokens;
    istringstream iss(line);
    string tok;
    while (iss >> tok) tokens.push_back(tok);
    return tokens;
}

static int count_leading_dots(const vector<string>& tokens) {
    int c = 0;
    for (const auto& t : tokens) {
        if (t == ".") ++c;
        else break;
    }
}
```



<CODEWARS2026/>
LEÓN

```
    }  
    return c;  
}  
static string build_number_from_tokens(const vector<string>& tokens,  
                                       const unordered_map<char, char>&  
mapping) {  
    string s;  
    for (const auto& t : tokens) {  
        if (t == ".") continue;  
        // Por especificación: t será '.' o una letra; aun así,  
        // si llega un dígito literal, lo copiamos por seguridad.  
        if (t.size() == 1 && isalpha(static_cast<unsigned char>(t[0]))) {  
            auto it = mapping.find(t[0]);  
            if (it == mapping.end()) {  
                // No debería ocurrir cuando el mapeo está completo;  
                // devolvemos vacío para indicar invalidez.  
                return "";  
            }  
            s.push_back(it->second);  
        } else {  
            // Copiamos token "tal cual" (podría ser un dígito literal)  
            s += t;  
        }  
    }  
    return s;  
}  
static vector<string> parse_input() {  
    vector<string> lines;
```




```
string line;
while (true) {
    if (!std::getline(cin, line)) break;
    // elimina trailing spaces
    while (!line.empty() && (line.back() == '\r' || line.back() == ' '
|| line.back() == '\t'))
        line.pop_back();
    if (line == ";") break;
    if (!line.empty()) lines.push_back(line); // ignorar líneas vacías
}
return lines;
}

static vector<char> extract_letters(const vector<string>& lines) {
    set<char> letters_set;
    for (const auto& line : lines) {
        for (unsigned char ch : line) {
            if (isalpha(ch)) letters_set.insert(static_cast<char>(ch));
        }
    }
    vector<char> letters(letters_set.begin(), letters_set.end());
    return letters;
}

static set<char> find_leading_letters(const vector<string>& lines) {
    set<char> leading;
    for (const auto& line : lines) {
        auto tokens = tokens_from_line(line);
        vector<string> relevant;
        for (const auto& t : tokens) if (t != ".") relevant.push_back(t);
    }
}
```

```
        if (relevant.size() <= 1) continue;
        for (const auto& t : tokens) {
            if (t == ".") continue;
            if (t.size() == 1 && isalpha(static_cast<unsigned
char>(t[0]))) {
                leading.insert(t[0]);
            }
            // si es dígito literal, no añadimos
            break;
        }
    }
    return leading;
}

// === Comprobación completa de un mapeo ===
static bool check_mapping(const unordered_map<char, char>& mapping,
                        const vector<vector<string>>& lines_tokens) {
    // Construye números de cada línea (ignorando las vacías)
    vector<long long> numbers;
    vector<string> numstr_per_line;
    numstr_per_line.reserve(lines_tokens.size());
    for (const auto& tokens : lines_tokens) {
        string numstr = build_number_from_tokens(tokens, mapping);
        numstr_per_line.push_back(numstr);
        if (numstr.empty()) {
            // línea sin números (posible), la ignoramos en 'numbers'
            continue;
        }
        if (numstr.size() > 1 && numstr[0] == '0') {
```



<CODEWARS2026/>
LEÓN

```
        // cero a la izquierda
        return false;
    }
    // Conversión a entero; asume que cabe en 64 bits
    // (si los casos reales exceden 18-19 dígitos, habría que usar
    enteros grandes por string).
    long long val = 0;
    for (char c : numstr) {
        if (!isdigit(static_cast<unsigned char>(c))) return false;
        int d = c - '0';
        // Chequeo simple de overflow (conservador)
        if (val > (LLONG_MAX - d) / 10) return false;
        val = val * 10 + d;
    }
    numbers.push_back(val);
}
if (numbers.size() < 3) return false;
long long multiplicand1 = numbers[0];
long long multiplicand2 = numbers[1];
long long final_result = numbers.back();
if (multiplicand1 * multiplicand2 != final_result) return false;
// === Verificación de productos parciales con alineado por puntos ===
int total_columns = static_cast<int>(lines_tokens.back().size());
if (total_columns <= 0) return false;
// parciales: lines_tokens[2 .. n-2]
vector<vector<string>> partial_lines;
if (lines_tokens.size() >= 4) { // al menos 2 factores + 1 parcial +
resultado
```



<CODEWARS2026/>
LEÓN

```
        for (size_t i = 2; i + 1 < lines_tokens.size(); ++i) {
            partial_lines.push_back(lines_tokens[i]);
        }
    }

    // reconstruir cadena del segundo factor (por tokens) para
    inspeccionar dígitos
    string m2_str = build_number_from_tokens(lines_tokens[1], mapping);
    if (m2_str.empty()) return false;
    // Debe haber tantos parciales como dígitos del segundo factor
    if (static_cast<int>(partial_lines.size()) !=
static_cast<int>(m2_str.size()))
        return false;
    // Verificar cada parcial i (0 = unidades)
    for (int i = 0; i < static_cast<int>(partial_lines.size()); ++i) {
        const auto& partial_tokens = partial_lines[i];
        int digit = m2_str[static_cast<int>(m2_str.size()) - 1 - i] - '0';
        long long expected_val = multiplicand1 * digit;
        string expected_value_str = to_string(expected_val);
        int expected_left_dots = total_columns -
(static_cast<int>(expected_value_str.size()) + i);
        int actual_left_dots = count_leading_dots(partial_tokens);
        string actual_numstr = build_number_from_tokens(partial_tokens,
mapping);
        if (digit == 0) {
            // Aceptamos: a) solo puntos; b) "0" alineado
            bool only_dots = true;
            for (const auto& t : partial_tokens) {
                if (t != ".") { only_dots = false; break; }
            }
        }
    }
}
```




<CODEWARS2026/>
LEÓN

```
    }
    if (only_dots) {
        continue; // válido como parcial 0 desplazado
    } else {
        if (!(actual_numstr == "0" && actual_left_dots ==
expected_left_dots)) {
            return false;
        }
    }
} else {
    if (!(actual_numstr == expected_value_str && actual_left_dots
== expected_left_dots)) {
        return false;
    }
}
return true;
}

// === Búsqueda por backtracking: asignar letras -> dígitos ===
struct Solver {
    const vector<string> lines;
    const vector<vector<string>> lines_tokens;
    const vector<char> letters;           // letras ordenadas
    const set<char> leading_letters;     // no pueden ser '0'
    // Estado de backtracking
    unordered_map<char, char> mapping;   // letra -> dígito (char
'0'..'9')
    array<bool, 10> used{};              // dígitos usados
};
```



<CODEWARS2026/>
LEÓN

```
Solver(vector<string> lines_)
: lines(lines_),
  lines_tokens([&](){
    vector<vector<string>> v; v.reserve(lines_.size());
    for (const auto& ln : lines_)
      v.push_back(tokens_from_line(ln));
    return v;
  }()),
  letters(extract_letters(lines_)),
  leading_letters(find_leading_letters(lines_)) {
  used.fill(false);
}

bool solve_all_assigned() {
  // Validar mapeo completo
  return check_mapping(mapping, lines_tokens);
}

bool dfs_assign(int idx) {
  if (idx == static_cast<int>(letters.size())) {
    return solve_all_assigned();
  }
  char L = letters[idx];
  for (int d = 0; d <= 9; ++d) {
    if (used[d]) continue;
    if (d == 0 && leading_letters.count(L)) continue; // poda:
    líderes no pueden ser 0
    mapping[L] = static_cast<char>('0' + d);
    used[d] = true;
    // (Opcional: se puede añadir poda adicional aquí)
```



<CODEWARS2026/>
LEÓN

```
        if (dfs_assign(idx + 1)) return true;
        used[d] = false;
        mapping.erase(L);
    }
    return false;
}

// Punto de entrada
bool run(unordered_map<char, char>& out_mapping) {
    if (letters.size() > 10) return false; // imposible
    bool ok = dfs_assign(0);
    if (ok) out_mapping = mapping;
    return ok;
}

};

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    vector<string> lines = parse_input();
    Solver solver(lines);
    unordered_map<char, char> mapping;
    if (solver.run(mapping)) {
        // imprimir en orden alfabético sin espacio final: A=6;B=3;...
        vector<char> keys = solver.letters; // ya vienen ordenadas
        string out;
        for (size_t i = 0; i < keys.size(); ++i) {
            char k = keys[i];
            out += k;
            out += '=';
        }
    }
}
```



<CODEWARS2026/>
LEÓN

```
        out += mapping[k];
        if (i + 1 < keys.size()) out += ' ';
    }
    cout << out << "\n";
} else {
    cout << "No solution found\n";
}
return 0;
}
```

Java

```
import java.io.*;
import java.math.BigInteger;
import java.util.*;
import java.util.stream.Collectors;

public class The_Enigmatic_Multiplication {
    // === Parseo y utilidades de tokens ===
    static List<String> parseInput(BufferedReader br) throws IOException {
        List<String> lines = new ArrayList<>();
        String line;
        while ((line = br.readLine()) != null) {
            line = rstrip(line);
            if (line.equals(";")) break;
            if (!line.isEmpty()) lines.add(line); // ignorar líneas vacías
        }
        return lines;
    }

    static String rstrip(String s) {
        int i = s.length() - 1;
```

<let's code/>





<CODEWARS2026/>
LEÓN

```
        while (i >= 0 && (s.charAt(i) == '\r' || s.charAt(i) == ' ' ||
s.charAt(i) == '\t')) i--;
        return s.substring(0, i + 1);
    }
    static List<String> tokensFromLine(String line) {
        if (line.isEmpty()) return Collections.emptyList();
        String[] parts = line.trim().split("\\s+");
        return Arrays.asList(parts);
    }
    static int countLeadingDots(List<String> tokens) {
        int c = 0;
        for (String t : tokens) {
            if (t.equals(".")) c++;
            else break;
        }
        return c;
    }
    static String buildNumberFromTokens(List<String> tokens,
Map<Character, Character> mapping) {
        StringBuilder sb = new StringBuilder();
        for (String t : tokens) {
            if (t.equals(".")) continue;
            if (t.length() == 1 && Character.isAlphabetic(t.charAt(0))) {
                Character mapped = mapping.get(t.charAt(0));
                if (mapped == null) {
                    // No debería pasar cuando mapping está completo;
devolvemos vacío como inválido
                    return "";
                }
            }
            sb.append(mapped);
        }
        return sb.toString();
    }
}
```



<CODEWARS2026/>
LEÓN

```
        }
        sb.append(mapped);
    } else {
        // Copiamos el token tal cual (si fuera dígito literal)
        sb.append(t);
    }
}
return sb.toString();
}

static List<Character> extractLetters(List<String> lines) {
    SortedSet<Character> set = new TreeSet<>();
    for (String line : lines) {
        for (int i = 0; i < line.length(); i++) {
            char ch = line.charAt(i);
            if (Character.isAlphabetic(ch)) set.add(ch);
        }
    }
    return new ArrayList<>(set);
}

static Set<Character> findLeadingLetters(List<String> lines) {
    Set<Character> leading = new HashSet<>();
    for (String line : lines) {
        List<String> tokens = tokensFromLine(line);
        List<String> relevant = tokens.stream().filter(t ->
!t.equals(".")).collect(Collectors.toList());
        if (relevant.size() <= 1) continue;
        for (String t : tokens) {
            if (t.equals(".")) continue;

```

```
        if (t.length() == 1 &&
Character.isAlphabetic(t.charAt(0))) {
            leading.add(t.charAt(0));
        }
        // si no es letra (por ejemplo, dígito literal), no
añadimos
        break;
    }
}
return leading;
}
// === Verificación completa para un mapeo ===
static boolean checkMapping(Map<Character, Character> mapping,
List<List<String>> linesTokens) {
    List<BigInteger> numbers = new ArrayList<>();
    List<String> numStrPerLine = new ArrayList<>(linesTokens.size());
    for (List<String> tokens : linesTokens) {
        String numstr = buildNumberFromTokens(tokens, mapping);
        numStrPerLine.add(numstr);
        if (numstr.isEmpty()) {
            // línea sin números (posible), ignoramos en 'numbers'
            continue;
        }
        if (numstr.length() > 1 && numstr.charAt(0) == '0') {
            // cero a la izquierda
            return false;
        }
    }
}
```

```
        if (!numstr.chars().allMatch(Character::isDigit)) return
false;

        numbers.add(new BigInteger(numstr));
    }
    if (numbers.size() < 3) return false;
    BigInteger multiplicand1 = numbers.get(0);
    BigInteger multiplicand2 = numbers.get(1);
    BigInteger finalResult = numbers.get(numbers.size() - 1);
    if (!multiplicand1.multiply(multiplicand2).equals(finalResult))
return false;

    // === Verificación de productos parciales con alineado por puntos
===

    int totalColumns = linesTokens.get(linesTokens.size() - 1).size();
    if (totalColumns <= 0) return false;
    List<List<String>> partialLines = new ArrayList<>();
    if (linesTokens.size() >= 4) { // 2 factores + >=1 parcial +
resultado
        for (int i = 2; i + 1 < linesTokens.size(); i++) {
            partialLines.add(linesTokens.get(i));
        }
    }
    String m2Str = buildNumberFromTokens(linesTokens.get(1), mapping);
    if (m2Str.isEmpty()) return false;
    if (partialLines.size() != m2Str.length()) return false;
    for (int i = 0; i < partialLines.size(); i++) {
        List<String> partialTokens = partialLines.get(i);
        int digit = Character.digit(m2Str.charAt(m2Str.length() - 1 -
i), 10);
```




<CODEWARS2026/>
LEÓN

```
        if (digit < 0) return false;
        String expectedValueStr =
multiplicand1.multiply(BigInteger.valueOf(digit)).toString();
        int expectedLeftDots = totalColumns -
(expectedValueStr.length() + i);
        int actualLeftDots = countLeadingDots(partialTokens);
        String actualNumStr = buildNumberFromTokens(partialTokens,
mapping);
        if (digit == 0) {
            boolean onlyDots = partialTokens.stream().allMatch(t ->
t.equals("."));
            if (onlyDots) {
                continue; // válido: parcial cero como "todo puntos"
            } else {
                if (!(actualNumStr.equals("0") && actualLeftDots ==
expectedLeftDots)) {
                    return false;
                }
            }
        } else {
            if (!(actualNumStr.equals(expectedValueStr) &&
actualLeftDots == expectedLeftDots)) {
                return false;
            }
        }
    }
    return true;
}
```

<let's code/>





<CODEWARS2026/>
LEÓN

```
// === Backtracking: asignar letras -> dígitos ===
static final class Solver {
    final List<String> lines;
    final List<List<String>> linesTokens;
    final List<Character> letters;    // ordenadas
    final Set<Character> leadingLetters;
    final Map<Character, Character> mapping = new HashMap<>();
    final boolean[] used = new boolean[10];
    Solver(List<String> lines) {
        this.lines = lines;
        this.linesTokens = new ArrayList<>();
        for (String ln : lines)
this.linesTokens.add(tokensFromLine(ln));
        this.letters = extractLetters(lines);
        this.leadingLetters = findLeadingLetters(lines);
    }
    boolean dfsAssign(int idx) {
        if (idx == letters.size()) {
            return checkMapping(mapping, linesTokens);
        }
        char L = letters.get(idx);
        for (int d = 0; d <= 9; d++) {
            if (used[d]) continue;
            if (d == 0 && leadingLetters.contains(L)) continue;
            mapping.put(L, (char) ('0' + d));
            used[d] = true;
            if (dfsAssign(idx + 1)) return true;
            used[d] = false;
        }
    }
}
```



<CODEWARS2026/>
LEÓN

```
        mapping.remove(L);
    }
    return false;
}

boolean run(Map<Character, Character> outMapping) {
    if (letters.size() > 10) return false;
    boolean ok = dfsAssign(0);
    if (ok) outMapping.putAll(mapping);
    return ok;
}

}

public static void main(String[] args) throws Exception {
    BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
    List<String> lines = parseInput(br);
    Solver solver = new Solver(lines);
    Map<Character, Character> mapping = new HashMap<>();
    if (solver.run(mapping)) {
        // imprimir en orden alfabético: A=6;B=3;...
        List<Character> keys = new ArrayList<>(solver.letters); // ya
ordenadas

        StringBuilder out = new StringBuilder();
        for (int i = 0; i < keys.size(); i++) {
            char k = keys.get(i);
            out.append(k).append('=').append(mapping.get(k));
            if (i + 1 < keys.size()) out.append(';');
        }
        System.out.println(out);
    }
}
```



<CODEWARS2026/>
LEÓN

```
    } else {  
        System.out.println("No solution found");  
    }  
}  
}
```


25

The eternal calendar

14 points

Introduction

The Forge of Chronos: The Creation of the Eternal Calendar

Long ago, before time had shape and the seasons were but a chaotic dance of the elements, the gods convened on Mount Aetherion, the sacred peak where the threads of existence were woven. The world was young, and mortals struggled to make sense of the cycles of light and shadow, of warmth and frost. It was then that Chronos, the primordial god of time, decreed that order must be brought to the chaos.

He summoned his three divine children to undertake this monumental task: Eirene, goddess of peace and balance; Helion, god of the blazing sun; and Seleneia, goddess of the silver moon. Together, they were entrusted with crafting a system that would guide mortals through the endless flow of time—a calendar that would forever mark the rhythm of life.

The Council of Seasons

The gods debated for seven days and nights beneath the great Celestial Oak, whose leaves shimmered with the stars. Each deity brought their perspective.

- Eirene spoke first: "Let the calendar be a tapestry of harmony. Each season should reflect balance between light and dark, growth and rest."
- Helion, his golden hair blazing like a summer sun, countered: "No! The calendar must honor the sun's power. Mortals should celebrate its rise and lament its fall."
- Seleneia, draped in a cloak of stars, whispered: "The moon, too, has its place. Its cycles have long guided tides and dreams. We must weave its phases into this creation."

But their visions clashed, and unity seemed impossible. To resolve this, Chronos ordered them to descend to the mortal world and seek inspiration.

The Trial of Mortals

Eirene journeyed to the lush meadows of Elara, where mortals tended crops and prayed for bountiful harvests. She witnessed how the changing seasons brought both prosperity and hardship, teaching her that balance was indeed the foundation of life.

Helion descended into the desert of Pyraxia, where mortals worshiped the sun as a giver of life and feared its absence. He realized the sun's journey across the sky could mark the passage of days and inspire celebration.

Seleneia roamed the shores of Aqualis, where fishermen guided their boats by the moonlight. She understood how the moon's phases mirrored the flow of time, its waxing and waning offering a cycle of hope and renewal.

The Forge of Chronos

Returning to Mount Aetherion, the siblings now worked together, their lessons from the mortal realm guiding their hands. They toiled in the Forge of Chronos, a mythical place where time itself flowed like molten silver.

- Eirene divided the year into four harmonious seasons, each reflecting the balance of growth and rest.
- Helion crafted the solar arc, a golden ring marking the days of the year as the sun journeyed through the sky.
- Seleneia inscribed the phases of the moon onto an obsidian disc, ensuring that its cycles would guide the months.

Chronos, with his immense power, bound these elements together into a single artifact: the *Eternal Calendar*. Its face shimmered with solar radiance, lunar grace, and seasonal wisdom, a perfect union of their divine wills. The Eternal calendar was created with the following logo:



The Calendar's Legacy

When the gods presented the Eternal Calendar to the mortals, they rejoiced. Farmers could now sow and reap with certainty, sailors could navigate by moonlight, and all could celebrate the changing seasons with festivals of light, harvest, and renewal.

From that day forward, the Eternal Calendar not only guided mortals but also became a bridge between gods and humanity, reminding all of the delicate balance that sustains existence.

Chronos, watching from his throne, smiled. Time had been given shape, and with it, the world could thrive in harmony. Thus, the Eternal Calendar became the greatest legacy of the gods, its creation an epic tale of discord, unity, and divine purpose.

Exercise

Write a program that receives as input a year and returns the calendar for given year.

Input

The program will simply receive a year. For instance:

2024

Output

And it will output that's years calendar. Something like:

2024																				
January							February							March						
Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa
	1	2	3	4	5	6					1	2	3						1	2
7	8	9	10	11	12	13	4	5	6	7	8	9	10	3	4	5	6	7	8	9
14	15	16	17	18	19	20	11	12	13	14	15	16	17	10	11	12	13	14	15	16
21	22	23	24	25	26	27	18	19	20	21	22	23	24	17	18	19	20	21	22	23
28	29	30	31				25	26	27	28	29			24	25	26	27	28	29	30
														31						
April							May							June						
Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa
	1	2	3	4	5	6					1	2	3							1
7	8	9	10	11	12	13	5	6	7	8	9	10	11	2	3	4	5	6	7	8
14	15	16	17	18	19	20	12	13	14	15	16	17	18	9	10	11	12	13	14	15
21	22	23	24	25	26	27	19	20	21	22	23	24	25	16	17	18	19	20	21	22
28	29	30					26	27	28	29	30	31		23	24	25	26	27	28	29
														30						
July							August							September						
Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa
	1	2	3	4	5	6					1	2	3	1	2	3	4	5	6	7
7	8	9	10	11	12	13	4	5	6	7	8	9	10	8	9	10	11	12	13	14
14	15	16	17	18	19	20	11	12	13	14	15	16	17	15	16	17	18	19	20	21
21	22	23	24	25	26	27	18	19	20	21	22	23	24	22	23	24	25	26	27	28
28	29	30	31				25	26	27	28	29	30	31	29	30					
October							November							December						
Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa
		1	2	3	4	5						1	2	1	2	3	4	5	6	7
6	7	8	9	10	11	12	3	4	5	6	7	8	9	8	9	10	11	12	13	14
13	14	15	16	17	18	19	10	11	12	13	14	15	16	15	16	17	18	19	20	21
20	21	22	23	24	25	26	17	18	19	20	21	22	23	22	23	24	25	26	27	28
27	28	29	30	31			24	25	26	27	28	29	30	29	30	31				

Example 1

Input

2024

Output

2024																				
January							February							March						
Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa
	1	2	3	4	5	6					1	2	3						1	2
7	8	9	10	11	12	13	4	5	6	7	8	9	10	3	4	5	6	7	8	9
14	15	16	17	18	19	20	11	12	13	14	15	16	17	10	11	12	13	14	15	16
21	22	23	24	25	26	27	18	19	20	21	22	23	24	17	18	19	20	21	22	23
28	29	30	31				25	26	27	28	29			24	25	26	27	28	29	30
														31						
April							May							June						
Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa
	1	2	3	4	5	6					1	2	3							1
7	8	9	10	11	12	13	5	6	7	8	9	10	11	2	3	4	5	6	7	8
14	15	16	17	18	19	20	12	13	14	15	16	17	18	9	10	11	12	13	14	15
21	22	23	24	25	26	27	19	20	21	22	23	24	25	16	17	18	19	20	21	22
28	29	30					26	27	28	29	30	31		23	24	25	26	27	28	29
														30						
July							August							September						
Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa
	1	2	3	4	5	6					1	2	3	1	2	3	4	5	6	7
7	8	9	10	11	12	13	4	5	6	7	8	9	10	8	9	10	11	12	13	14
14	15	16	17	18	19	20	11	12	13	14	15	16	17	15	16	17	18	19	20	21
21	22	23	24	25	26	27	18	19	20	21	22	23	24	22	23	24	25	26	27	28
28	29	30	31				25	26	27	28	29	30	31	29	30					
October							November							December						
Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa
		1	2	3	4	5						1	2	1	2	3	4	5	6	7
6	7	8	9	10	11	12	3	4	5	6	7	8	9	8	9	10	11	12	13	14
13	14	15	16	17	18	19	10	11	12	13	14	15	16	15	16	17	18	19	20	21
20	21	22	23	24	25	26	17	18	19	20	21	22	23	22	23	24	25	26	27	28
27	28	29	30	31			24	25	26	27	28	29	30	29	30	31				

Solutions

Python

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import sys
# Parámetros globales (equivalentes a las globals del C)
```



<CODEWARS2026/> LEÓN

```
width = 80
year = None
cols = 0
lead = 0
gap = 0
wdays = ["Su", "Mo", "Tu", "We", "Th", "Fr", "Sa"]
# months: name, days, start_wday, at
months = [
    {"name": "January",    "days": 31, "start_wday": 0, "at": 0},
    {"name": "February",  "days": 28, "start_wday": 0, "at": 0},
    {"name": "March",     "days": 31, "start_wday": 0, "at": 0},
    {"name": "April",     "days": 30, "start_wday": 0, "at": 0},
    {"name": "May",       "days": 31, "start_wday": 0, "at": 0},
    {"name": "June",      "days": 30, "start_wday": 0, "at": 0},
    {"name": "July",      "days": 31, "start_wday": 0, "at": 0},
    {"name": "August",    "days": 31, "start_wday": 0, "at": 0},
    {"name": "September", "days": 30, "start_wday": 0, "at": 0},
    {"name": "October",   "days": 31, "start_wday": 0, "at": 0},
    {"name": "November",  "days": 30, "start_wday": 0, "at": 0},
    {"name": "December",  "days": 31, "start_wday": 0, "at": 0},
]

def space(n: int) -> None:
    """Imprime n espacios."""
    if n > 0:
        sys.stdout.write(" " * n)

def init_months() -> None:
    """Inicializa días de febrero si bisiesto, inicio de semana por mes y
    layout."""
```



<CODEWARS2026/> LEÓN

```
global year, cols, gap, lead, months

# Ajuste por año bisiesto: (divisible por 4 y no por 100) o divisible
por 400
if ((year % 4 == 0 and year % 100 != 0) or (year % 400 == 0)):
    months[1]["days"] = 29
else:
    months[1]["days"] = 28

# Cálculo del día de la semana del 1 de enero (Sunday=0)
y = year - 1
months[0]["start_wday"] = (y * 365 + y // 4 - y // 100 + y // 400 + 1)
% 7

# Inicio de semana del resto de meses
for i in range(1, 12):
    months[i]["start_wday"] = (months[i - 1]["start_wday"] + months[i
- 1]["days"]) % 7

# Configurar columnas según ancho
cols = (width + 2) // 22
while 12 % cols != 0:
    cols -= 1

gap = ((width - 20 * cols) // (cols - 1)) if (cols - 1) > 0 else 0
if gap > 4:
    gap = 4

lead = (width - (20 + gap) * cols + gap + 1) // 2

def print_row(row: int) -> None:
    """Imprime una fila de 'cols' meses con encabezado y semanas."""
    global cols, lead, gap, months
    from_idx = row * cols
    to_idx = from_idx + cols
```



<CODEWARS2026/>
LEÓN

```
# Títulos de los meses (centrados en 20)
space(lead)
for c in range(from_idx, to_idx):
    name = months[c]["name"]
    i = len(name)
    left = (20 - i) // 2
    space(left)
    sys.stdout.write(name)
    # Relleno derecho + gap entre meses (no añadir gap al último)
    right = 20 - i - left + (0 if c == to_idx - 1 else gap)
    space(right)
sys.stdout.write("\n")
# Encabezado de días
space(lead)
for c in range(from_idx, to_idx):
    for i in range(7):
        sys.stdout.write(wdays[i])
        if i != 6:
            sys.stdout.write(" ")
    if c < to_idx - 1:
        space(gap)
    else:
        sys.stdout.write("\n")
# Semanas (días)
while True:
    # Comprobar si quedan días por imprimir en esta fila
    any_remaining = False
    for c in range(from_idx, to_idx):
```




<CODEWARS2026/>
LEÓN

```
        if months[c]["at"] < months[c]["days"]:
            any_remaining = True
            break
    if not any_remaining:
        break
    space(lead)
    for c in range(from_idx, to_idx):
        i = 0
        # Desplazamiento inicial según start_wday
        for _ in range(months[c]["start_wday"]):
            space(3)
            i += 1
        # Imprimir hasta completar la semana o agotar días del mes
        while i < 7 and months[c]["at"] < months[c]["days"]:
            months[c]["at"] += 1
            sys.stdout.write(f"{months[c]['at']:2d}")
            i += 1
            # Espacio entre días o entre meses
            if i < 7 or c < to_idx - 1:
                sys.stdout.write(" ")
        # Relleno hasta completar los 7 días visibles de la línea para
este mes
        while i < 7 and c < to_idx - 1:
            space(3)
            i += 1
        # Espaciado extra entre meses (ajustado)
        if c < to_idx - 1:
            space(gap - 1)
```



<CODEWARS2026/> LEÓN

```
# Tras la primera línea de semanas, el desplazamiento se anula
months[c]["start_wday"] = 0
sys.stdout.write("\n")
# Línea en blanco entre filas (excepto la última con diciembre)
if from_idx + cols < 12:
    sys.stdout.write("\n")
def print_year() -> None:
    """Imprime el año centrado y todas las filas de meses."""
    global year, width, cols
    buf = str(year)
    space((width - len(buf)) // 2)
    sys.stdout.write(buf + "\n\n")
    row = 0
    while row * cols < 12:
        print_row(row)
        row += 1
def bail(argv0: str) -> None:
    """Imprime uso y sale con error."""
    sys.stderr.write(f"bad args\nUsage: {argv0} year [-w width (>=
20)]\n")
    sys.exit(1)
def main() -> None:
    global width, year, months
    argv = sys.argv
    argc = len(argv)
    # Si hay año por argumento, úsalo; si no, intenta leer de stdin.
    year_set = False
    # Parseo de argumentos al estilo del C
```



<CODEWARS2026/> LEÓN

```
i = 1
while i < argc:
    if argv[i] == "-w":
        i += 1
        if i == argc:
            bail(argv[0])
        try:
            width = int(argv[i])
        except ValueError:
            bail(argv[0])
        if width < 20:
            bail(argv[0])
    else:
        # Primer argumento no -w lo tratamos como año
        if not year_set:
            try:
                y = int(argv[i])
            except ValueError:
                y = 1969
            if y <= 0:
                y = 1969
            year = y
            year_set = True
        else:
            bail(argv[0])
    i += 1
# Si no se estableció año por argv, intenta leer de stdin
if not year_set:
```



<CODEWARS2026/> LEÓN

```
try:
    line = sys.stdin.readline()
    if line:
        y = int(line.strip())
        if y <= 0:
            y = 1969
        year = y
    else:
        # Sin entrada: usar valor por defecto del C
        year = 1969
except Exception:
    year = 1969

# Seguridad: reiniciar 'at' por si se reutiliza el módulo
for m in months:
    m["at"] = 0
init_months()
print_year()

if __name__ == "__main__":
    main()
```

C++

```
//#Source: shorturl.at/adg05
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int width = 80, year;
int cols, lead, gap;
const char *wdays[] = { "Su", "Mo", "Tu", "We", "Th", "Fr", "Sa" };
struct months {
```




<CODEWARS2026/> LEÓN

```
const char *name;
int days, start_wday, at;
} months[12] = {
    { "January",    31, 0, 0 },
    { "February",   28, 0, 0 },
    { "March",      31, 0, 0 },
    { "April",      30, 0, 0 },
    { "May",        31, 0, 0 },
    { "June",       30, 0, 0 },
    { "July",       31, 0, 0 },
    { "August",     31, 0, 0 },
    { "September",  30, 0, 0 },
    { "October",    31, 0, 0 },
    { "November",   30, 0, 0 },
    { "December",   31, 0, 0 }
};

void space(int n) { while (n-- > 0) putchar(' '); }
void init_months()
{
    int i;
    if (!(year % 4) && (year % 100) || !(year % 400))
        months[1].days = 29;
    year--;
    months[0].start_wday
        = (year * 365 + year/4 - year/100 + year/400 + 1) % 7;
    for (i = 1; i < 12; i++)
        months[i].start_wday =
            (months[i-1].start_wday + months[i-1].days) % 7;
```



<CODEWARS2026/>
LEÓN

```
cols = (width + 2) / 22;
while (12 % cols) cols--;
gap = cols - 1 ? (width - 20 * cols) / (cols - 1) : 0;
if (gap > 4) gap = 4;
lead = (width - (20 + gap) * cols + gap + 1) / 2;
    year++;
}
void print_row(int row)
{
    int c, i, from = row * cols, to = from + cols;
    space(lead);
    for (c = from; c < to; c++) {
        i = strlen(months[c].name);
        space((20 - i)/2);
        printf("%s", months[c].name);
        space(20 - i - (20 - i)/2 + ((c == to - 1) ? 0 : gap));
    }
    putchar('\n');
    space(lead);
    for (c = from; c < to; c++) {
        for (i = 0; i < 7; i++)
            printf("%s%s", wdays[i], i == 6 ? "" : " ");
        if (c < to - 1) space(gap);
        else putchar('\n');
    }
    while (1) {
        for (c = from; c < to; c++)
            if (months[c].at < months[c].days) break;
    }
}
```



<CODEWARS2026/>
LEÓN

```
        if (c == to) break;
        space(lead);
        for (c = from; c < to; c++) {
            for (i = 0; i < months[c].start_wday; i++) space(3);
            while(i++ < 7 && months[c].at < months[c].days) {
                printf("%2d", ++months[c].at);
                if (i < 7 || c < to - 1) putchar(' ');
            }
            while (i++ <= 7 && c < to - 1) space(3);
            if (c < to - 1) space(gap - 1);
            months[c].start_wday = 0;
        }
        putchar('\n');
    }
    // Check if it's the last row (with December) and don't add an extra
    line break.
    if (from + cols < 12) {
        putchar('\n');
    }
}

void print_year()
{
    int row;
    char buf[32];
    sprintf(buf, "%d", year);
    space((width - strlen(buf)) / 2);
    printf("%s\n\n", buf);
    for (row = 0; row * cols < 12; row++)
```



<CODEWARS2026/>
LEÓN

```
        print_row(row);
    }
    int main(int c, char **v)
    {
        int i, year_set = 0;
        scanf("%d", &year);
        for (i = 1; i < c; i++) {
            if (!strcmp(v[i], "-w")) {
                if (++i == c || (width = atoi(v[i])) < 20)
                    goto bail;
            } else if (!year_set) {
                if (!sscanf(v[i], "%d", &year) || year <= 0)
                    year = 1969;
                year_set = 1;
            } else
                goto bail;
        }
        init_months();
        print_year();
        return 0;
    bail: fprintf(stderr, "bad args\nUsage: %s year [-w width (>= 20)]\n",
v[0]);
        exit(1);
    }
}
```

Java

```
import java.util.Scanner;
public class Main {
    // Parámetros globales (equivalentes a las globals del C)
```




<CODEWARS2026/> LEÓN

```
static int width = 80, year;
static int cols, lead, gap;
static final String[] wdays = { "Su", "Mo", "Tu", "We", "Th", "Fr",
"Sa" };
static class Month {
    final String name;
    int days;
    int startWday;
    int at; // contador de días impresos
    Month(String name, int days) {
        this.name = name;
        this.days = days;
        this.startWday = 0;
        this.at = 0;
    }
}
static final Month[] months = {
    new Month("January", 31),
    new Month("February", 28),
    new Month("March", 31),
    new Month("April", 30),
    new Month("May", 31),
    new Month("June", 30),
    new Month("July", 31),
    new Month("August", 31),
    new Month("September", 30),
    new Month("October", 31),
    new Month("November", 30),
```



<CODEWARS2026/>
LEÓN

```
        new Month("December", 31)
    };
    static void space(int n) {
        if (n <= 0) return;
        for (int i = 0; i < n; i++) System.out.print(' ');
    }
    static void initMonths() {
        // Año bisiesto: (divisible por 4 y no por 100) o divisible por
400
        if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0))
            months[1].days = 29;
        else
            months[1].days = 28;
        // Día de la semana del 1 de enero (Sunday=0) usando el mismo
cómputo que el C
        int y = year - 1;
        months[0].startWday = (y * 365 + y / 4 - y / 100 + y / 400 + 1) %
7;

        // Inicio de semana del resto de meses (cascada)
        for (int i = 1; i < 12; i++) {
            months[i].startWday = (months[i - 1].startWday + months[i -
1].days) % 7;
        }
        // Configurar columnas según ancho (replica el C)
        cols = (width + 2) / 22;
        while (12 % cols != 0) cols--;
        gap = (cols - 1) > 0 ? (width - 20 * cols) / (cols - 1) : 0;
        if (gap > 4) gap = 4;
```



<CODEWARS2026/>
LEÓN

```
        lead = (width - (20 + gap) * cols + gap + 1) / 2;
        // Reiniciar contadores "at" por seguridad
        for (Month m : months) m.at = 0;
    }
    static void printRow(int row) {
        int from = row * cols;
        int to = from + cols;
        // Títulos de los meses (centrados en 20)
        space(lead);
        for (int c = from; c < to; c++) {
            String name = months[c].name;
            int i = name.length();
            int left = (20 - i) / 2;
            space(left);
            System.out.print(name);
            int right = 20 - i - left + ((c == to - 1) ? 0 : gap);
            space(right);
        }
        System.out.println();
        // Encabezado de días
        space(lead);
        for (int c = from; c < to; c++) {
            for (int i = 0; i < 7; i++) {
                System.out.print(wdays[i]);
                if (i != 6) System.out.print(' ');
            }
            if (c < to - 1) space(gap);
            else System.out.println();
        }
    }
}
```



<CODEWARS2026/>
LEÓN

```
    }  
    // Semanas (días)  
    while (true) {  
        boolean anyRemaining = false;  
        for (int c = from; c < to; c++) {  
            if (months[c].at < months[c].days) { anyRemaining = true;  
break; }  
        }  
        if (!anyRemaining) break;  
        space(lead);  
        for (int c = from; c < to; c++) {  
            int i = 0;  
            // Desplazamiento inicial según startWday  
            for (int s = 0; s < months[c].startWday; s++) {  
                space(3);  
                i++;  
            }  
            // Imprimir hasta completar la semana o agotar días del  
mes  
            while (i < 7 && months[c].at < months[c].days) {  
                months[c].at++;  
                System.out.printf("%2d", months[c].at);  
                i++;  
                // Espacio entre días o entre meses  
                if (i < 7 || c < to - 1) System.out.print(' ');  
            }  
            // Relleno hasta completar los 7 "slots" visibles de la  
línea para este mes
```




<CODEWARS2026/>
LEÓN

```
        while (i < 7 && c < to - 1) {
            space(3);
            i++;
        }
        // Espaciado extra entre meses (ajustado)
        if (c < to - 1) space(gap - 1);
        // Tras la primera línea del mes, anular desplazamiento
        months[c].startWday = 0;
    }
    System.out.println();
}
// Línea en blanco entre filas (excepto la última con diciembre)
if (from + cols < 12) System.out.println();
}
static void printYear() {
    String buf = Integer.toString(year);
    space((width - buf.length()) / 2);
    System.out.println(buf);
    System.out.println();
    for (int row = 0; row * cols < 12; row++) {
        printRow(row);
    }
}
static void bail(String prog) {
    System.err.printf("bad args%nUsage: %s year [-w width (>= 20)]%n",
prog);
    System.exit(1);
}
```



```
public static void main(String[] args) {  
    String prog = Main.class.getSimpleName();  
    // Primero: intentar leer 'year' de stdin como en el C  
    Integer stdinYear = null;  
    Scanner sc = new Scanner(System.in);  
    if (sc.hasNextInt()) {  
        stdinYear = sc.nextInt();  
    }  
    boolean yearSet = false;  
    // Parseo de argumentos al estilo del C  
    for (int i = 0; i < args.length; i++) {  
        if ("-w".equals(args[i])) {  
            i++;  
            if (i == args.length) bail(prog);  
            try {  
                width = Integer.parseInt(args[i]);  
            } catch (NumberFormatException e) {  
                bail(prog);  
            }  
            if (width < 20) bail(prog);  
        } else if (!yearSet) {  
            int y;  
            try {  
                y = Integer.parseInt(args[i]);  
            } catch (NumberFormatException e) {  
                y = 1969;  
            }  
            if (y <= 0) y = 1969;  
        }  
    }  
}
```



<CODEWARS2026/>
LEÓN

```
        year = y;
        yearSet = true;
    } else {
        bail(prog);
    }
}

// Si no se estableció año por argv, usar el leído de stdin; si no
hay, usar 1969
if (!yearSet) {
    if (stdinYear != null && stdinYear > 0) {
        year = stdinYear;
    } else {
        year = 1969;
    }
}
initMonths();
printYear();
}
}
```

<let's code/>

