CODEWARS
LEÓN - MADRID
2023

CodeWars solutions 2023

SCDS

Toda la información en:
hpscds.com/codewars2023/

1 Welcome to Codewars

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 2023!".

Exercise

Write a program that outputs the string "Welcome to Codewars 2023!".

Input

None.

Output

The string "Welcome to Codewars 2023!".

Example 1

Input

No input for this problem.

Output

```
Welcome to Codewars 2023!
```

Solutions

Python

```
def main():
    print("Welcome to Codewars 2023!")
if __name__ == "__main__":
    main()
```

C++

```
#include <iostream>
int main(void) {
    std::cout << "Welcome to Codewars 2023!" << std::endl;
    return 0;
}
```

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 2023, TeamName!*, where *TeamName* is the name of the team you input.

Exercise

Write a program that outputs the string *Welcome to Codewars 2023, 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 2023, TeamName!* where *TeamName* is the name of the team you input.

Example 1

Input

```
Trambolikos
```

Output

```
Welcome to Codewars 2023, Trambolikos!
```

Example 2

Input

```
Los top del Mundo
```

Output

```
Welcome to Codewars 2023, Los top del Mundo!
```

Solutions

Python

```
def main():
    team = input()
    print("Welcome to Codewars 2023, " + 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 2023, " << team << "!" << std::endl;
    return 0;
}
```

3 3TPG yb srawedoC ot emocleW

4 points

Introduction

DISCLAIMER: This problem has been written completely by GPT3, a model language that for sure you have heard about lately.

You are a member of an archeological team that's exploring a dense, wild forest in search of ancient ruins. One of the ruins you discover has inscriptions on the walls, but the inscriptions are in an ancient language and written in reverse order.

Exercise

Your task is to write a program that takes in the encoded inscription as a string, and then prints out the original text in the correct order.

Input

A single string, representing the encoded inscription on the ancient ruins.

Output

A single string, representing the inscription in the correct order.

Example 1

Input

```
!3TPG yb ,3202 sraWedoC ot emocleW
```

Output

```
Welcome to CodeWars 2023, by GPT3!
```

Example 2

Input

```
...terces neddih a seil sniur eseht nI
```

Output

```
In these ruins lies a hidden secret...
```

Solutions

Python

```python
def main():
    print(input()[::-1])
if __name__ == "__main__":
    main()
```

C++

```cpp
#include <iostream>
#include <string>
std::string reverseString(std::string str) {
    int n = str.length();
    for (int i = 0; i < n / 2; i++) {
        std::swap(str[i], str[n - i - 1]);
    }
    return str;
}
int main() {
    std::string str;
    std::getline(std::cin, str);
    std::cout << reverseString(str) << std::endl;
    return 0;
}
```

4 Literal Acronyms

5 points

Introduction

An alien race called the Literalites has arrived to Earth. They had no trouble learning our languages, however they have the "problem" that they always interpret everything in a literal sense. So, whenever they see one of our acronyms that include a number and then some letters, they think its meaning is to repeat the letters as many times as the precedent number states. For example, when they see "4K" (for the resolution), they think it means "KKKK". And when they see 5G (for the mobile network speed), they think it means "GGGGG". Same for other acronyms like "1080p" (for number of lines in a screen) or "30fps" (for frames per second).

Exercise

You are asked to write a program that takes one of those acronyms and outputs what the aliens will interpret. See examples below.

Input

An acronym like "720p" or "3G".

Output

The interpretation by the aliens. For example, for "3G" it would be "GGG".

Example 1

Input

```
4K
```

Output

```
KKKK
```

Example 2

Input

```
1080p
```

Output

```
pppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
pppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
pppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
```

```
ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
pp
```

Example 3

Input

```
30fps
```

Output

```
fpsfpsfpsfpsfpsfpsfpsfpsfpsfpsfpsfpsfpsfpsfpsfpsfpsfpsfpsfpsfpsfpsfpsfpsfpsfp
sfpsfpsfpsfps
```

Solutions

Python

```
import re
def main():
    match = re.match(r'(?P<number>\d+)(?P<string>\w+)', input())
    if match is None:
        print('Error: invalid input')
        return
    times = int(match.group('number'))
    string = match.group('string')
    print(string*times)
if __name__ == '__main__':
    main()
```

C++

```
#include <iostream>
#include <regex>
int main()
{
    std::string input;
    std::getline(std::cin, input);
    std::regex pattern(R"((\d+)(\w+))");
    std::smatch match;
    if (!std::regex_match(input, match, pattern))
    {
        std::cout << "Error: invalid input" << std::endl;
        return 0;
    }
    int times = std::stoi(match.str(1));
    std::string string = match.str(2);
    for (int i = 0; i < times; ++i)
    {
        std::cout << string;
    }
    std::cout << std::endl;
    return 0;
}
```

5 The Neptune moons

5 points

Introduction

Neptune has 14 satellites, but almost nobody knows the order of proximity to the planet because, among other reasons, their orbital excentricities are very different and the order changes depending on the orbital location. It's generally agreed that the order of the fourteen Neptune moons (starting by the closest one) is Naiad, Thalassa, Despina, Galatea, Larissa, Hippocamp, Proteus, Triton, Nereid, Halimede, Sao, Laomedeia, Psamathe and Neso.

Exercise

You are asked to write a program to return the name of a moon given its order number.

Input

The input will be a number from 1 to 14.

Output

If the input is not a number, or is not a number from 1 to 14, the program will return

ERROR

Otherwise it will return the name of the moon, as written in the Introduction.

Example 1

Input

```
4
```

Output

```
Galatea
```

Example 2

Input

```
9
```

Output

```
Nereid
```

Example 3

Input

```
99
```

Output

```
ERROR
```

Solutions

Python

```
import sys
def main(argv):
   nNumero=int(input(""))
   moons=["Naiad","Thalassa", "Despina", "Galatea", "Larissa", "Hippocamp",
"Proteus", "Triton", "Nereid", "Halimede", "Sao", "Laomedeia",
"Psamathe","Neso"];
   if (nNumero<1):
      print("ERROR")
   elif (nNumero>14):
      print("ERROR")
   else:
      print(moons[nNumero-1])
if __name__ == "__main__":
   main(sys.argv)
```

C++

```
#include <string>
#include <iostream>
#include <vector>
#include <sstream>
int main(int argc, char **argv)
{
    std::string moons[14]={"Naiad","Thalassa", "Despina", "Galatea",
"Larissa", "Hippocamp", "Proteus", "Triton", "Nereid", "Halimede", "Sao",
"Laomedeia", "Psamathe","Neso"};
    std::string input_text="";
    //INPUT
    std::cin >> input_text;
    int input_val=std::stoi(input_text);
    if (input_val<=0 || input_val>14)
        std::cout<<"ERROR";
    else
        std::cout<<moons[input_val-1];
    return 0;
}
```

6 DNI letter

6 points

Introduction

You are new in a company that's developing a web interface for the the Spanish government.

In order to authenticate, users need to type their DNI numbers with the appropiate letter.

Your boss has asked you to write a program to check the letter typed by users is correct.

Exercise

To calculate the letter associated to the number, the number has to be divided by 23.

The remainder on this calculation indicates an index in a list of letters, in which you can get the associated letter.

This is the list of letters:

```
TRWAGMYFPDXBNJZSQVHLCKE
```

Given an ID number, generate the complete number with the associated letter.

Input

A positive number (normally 7 or 8 digits long, but there are shorter ones).

Output

The output will be the letter for the DNI corresponding to the number. Letter will be in uppercase, like in the list above.

Examples

Test the numbers in your DNIs. They should work.

Solutions

Python

```
import sys
def main(argv):
   nNumero=int(input(""))
   cDniLetters = "TRWAGMYFPDXBNJZSQVHLCKE";
   nLetterIndex = nNumero % 23;
   print(cDniLetters[nLetterIndex%23])
if __name__ == "__main__":
   main(sys.argv)
```

C++

```
#include <iostream>
#include <string>
using namespace std;
/*
  ::   .::::::::::::.
 ,;;   ;;,`;;;```.;;;
,[[[,,,[[[ `]]nnn]]'
"$$$""$$$  $$$""
 888   "88o 888o
 MMM    YMM YMMMb
  .,-:::::     ...    :::::::-.  .,::::::      .::    .   .::::::.
::::::::..   .::::::.
,;;;'````'  .;;;;;;;.  ;;,   `';,;;;;;''''      ';;,  ;;  ;;;' ;;`;;
;;;;``;;;; ;;;`    `
[[[        ,[[     \[[,`[[     [[ [[cccc        '[[, [[, [[' ,[[ '[[,
[[[,/[[[' '[==/[[[[,
`88bo,__,o,"888,_ _,88P 888_,o8P' 888oo,__         "88"888   888   888,888b
"88bo,88b    dP
  "YUMMMMMP" "YMMMMMP"  MMMMP"`   """"YUMMM         "M "M"   YMM   ""` MMMM
"W"  "YMmMY"

Calculate Spanish DNI number
To calculate the letter associated to the number, the number has to be
divided by 23
The rest on this calculation indicates a index in a list of letters, in which
you can get the associated letter
This is the list of letters : TRWAGMYFPDXBNJZSQVHLCKE
Given an ID number, generate the complete number with the associated leter
*/
```

```
int main()
{
    int nNumero;
    cin >> nNumero;
    //
    string cDniLetters = "TRWAGMYFPDXBNJZSQVHLCKE";
    int nLetterIndex = nNumero % 23;
    //cout << nNumero;
    cout << cDniLetters.substr( nLetterIndex , 1 );
    cout << "\r\n";
}
```

7 Armstrong

7 points

Introduction

An Armstrong number is an n-digit number that is equal to the sum of the nth powers of its individual digits.

For example, 153 is an Armstrong number because it has 3 digits and 1^3 + 5^3 + 3^3 = 153

Exercise

You are asked to write a program to test if a given number is an Armstrong number

To check whether any positive number is an Armstrong number or not, following is the example:

```
Since 153 is equal to 1*1*1 + 5*5*5 + 3*3*3. So 153 is an Armstrong number.
Since 12 is not equal to 1*1 + 2*2 (equals 5). So 12 is not an Armstrong
number.
```

Input

Input will consist in a positive number (greater or equal to zero).

Output

If the introduced number is an Armstrong number the output will be:

```
Armstrong!!
```

Otherwise the output will be:

```
No Armstrong :(
```

Example 1

Input

```
153
```

Output

```
Armstrong!!
```

Example 2

Input

```
421
```

Output

```
No Armstrong :(
```

Example 3

Input

```
0
```

Output

```
Armstrong!!
```

0 is an Armstrong number because 0^1 = 0.## Solutions

Python

```
# Author: Adrian Fernandez Alvarez (adrian.fernandez.alvarez@hp.com)
# Returns the number of digits of an integer
def getNumberOfDigits(n):
    digits = 0
    number = n
    while number > 0:
        number = int(number / 10)
        digits += 1
    return digits
# Returns true if the passed integer is an Armstrong Number
def isArmstrong(n):
    number = n
    digits = getNumberOfDigits(number)
    res = 0
    for i in range(n):
        res += pow(number % 10, digits)
        number = int(number / 10)
    return n == 0 or res == n
if __name__ == "__main__":
    number = input()
    try:
        number = int(number)
    except:
        print("Please, enter a valid integer")
        exit()
    if isArmstrong(number):
        print("Armstrong!!")
    else:
        print("No Armstrong :(")
```

C++

```
#include <iostream>
#include <string>
#include <cmath>
using namespace std;
/*
  ::   .:::::::::::.
 ,;;   ;;,`;;;```.;;;
,[[[,,,[[[ `]]nnn]]'
"$$$""""$$$  $$$""
 888   "88o 888o
 MMM    YMM YMMMb
  .,-:::::     ...    :::::::-.  .,::::::      .::    .   .::::::.
:::::::..   .::::::.
,;;;'````'  .;;;;;;;.  ;;,   `';,;;;;''''      ';;,  ;;  ;;;' ;;`;;
;;;;``;;;; ;;;`    `
[[[        ,[[     \[[,`[[     [[ [[cccc       '[[, [[, [[' ,[[ '[[,
[[[,/[[[' '[==/[[[[,
`88bo,__,o,"888,_ _,88P 888_,o8P' 888oo,__         "88"888   888   888,888b
"88bo,88b    dP
  "YUMMMMMP" "YMMMMMP"  MMMMP"`   """"YUMMM         "M "M"   YMM   ""` MMMM
"W"  "YMmMY"

Test if a given number is an Armstrong number
To check whether any positive number is an Armstrong number or not, following
is the example:
```

```
    Since 153 is equal to 1*1*1 + 5*5*5 + 3*3*3. So 153 is an Armstrong
number
    Since 12 is not equal to 1*1 + 2*2. So 12 is not an Armstrong number
*/
int getNumberOfDigits( int n )
{
    int digits = 0;
    int number = n;
    while (number > 0)
    {
    number /= 10;
        digits++;
    }
    return digits;
}
bool isArmstrong( int nNumber )
{
    int n, rem, num=0;
    n = nNumber;
    int digits = getNumberOfDigits( n );
    while( n !=0 )
    {
         rem = n%10;
         num = num + pow(rem, digits);
         n = n / 10;
    }
    return nNumber == 0 || num == nNumber;
}
int main()
{
    int nNumber;
    cin >> nNumber;
    // Test Armstrong
    if( isArmstrong( nNumber ) )
      cout << "Armstrong!!";
    else
      cout << "No Armstrong :(";
}
```

8 Prime powers

7 points

Introduction

The On-Line Encyclopedia of Integer Sequences (OEIS) defines the sequence Prime powers as “numbers of the form p^k where p is a prime and k >= 1.”

In this sequence 1 is not considered prime.

Exercise

Write a program that, given an integer value M, shows a list of all the prime powers lower than M, printing them sorted increasingly, separated by a white space.

Input

A single integer which is the upper limit to calculate prime powers.

Output

The list of prime powers below M which are prime powers.

Example 1

Input

```
45
```

Output

```
2 3 4 5 7 8 9 11 13 16 17 19 23 25 27 29 31 32 37 41 43
```

Example 2

Input

```
25
```

Output

```
2 3 4 5 7 8 9 11 13 16 17 19 23
```

Solutions

Python

```
#! /usr/bin/python3
def is_prime(N):
```

```python
    for i in range(2, N):
        if N % i == 0:
            return False
    return True
def main():
    lim = int(input())
    prime_powers = []
    for num in range(2, lim):
        if is_prime(num):
            # print(f"{num=}")
            pow_prime = num
            while pow_prime < lim:
                prime_powers.append(pow_prime)
                pow_prime *= num
                # print(f"{pow_prime=}")
    prime_powers.sort()
    # Print the list of primes separated by a white space
    print(*prime_powers, sep=' ')
if __name__ == '__main__':
    main()
```

C++

```cpp
#include <iostream>
#include <set>
using I = unsigned int;
bool is_prime(I num)
{
    for (I i = 2; i < num; i++)
    {
        if (num % i == 0)
            return false;
    }
    return true;
}
int main()
```

```
{
    I lim;
    std::cin >> lim;
    std::set<I> prime_powers;
    for (I num = 2; num < lim; num++)
    {
        if (is_prime(num))
        {
            auto pow_power = num;
            while (pow_power < lim)
            {
                prime_powers.insert(pow_power);
                pow_power *= num;
            }
        }
    }
    auto it = prime_powers.begin();
    std::cout << *it++;
    for (; it != prime_powers.end(); ++it)
    {
        std::cout << " " << *it;
    }
    return 0;
}
```

9 All Equal

9 points

Introduction

The year is 2782. The population in Spain has reached a new record of 1000 million people. As the country cannot support more population, the Spanish government has devised a new law: all people with the same digits in their ID number will be executed. However, in order to make it even more unfair, only those whose ID is divisible by an specific number will be included. For example, if the number is 37, only those whose ID has all the numbers equal (like 999) and is divisible by 37 will be executed.

The IDs for the people start in number 0 and end in 999 999 999. However, the 0 ID is excluded because it is reserved for the president of Spain and it is divisible by any number, having all the digits equal (just the 0). And you know, the game is rigged.

Exercise

You are asked to write a program that takes a number as a parameter and prints all the numbers lower than 1000 million that are (wholy) divisible by that number and have all the digits equal. in example, for the number 37, the program should print:

```
111;222;333;444;555;666;777;888;999;111111;222222;333333;444444;555555;666666
;777777;888888;999999;111111111;222222222;333333333;444444444;555555555;66666
6666;777777777;888888888;999999999
```

Because those are the numbers with all the digits equal and divisible by 37.

Input

A single integer that indicates the number to divide by.

Output

The list of IDs that are divisible by the number and have all the digits equal. Well, the list of people to be executed if you want to be more dramatic. They will be separated by the character *;* and ordered from the lowest to the highest.

Example 1

Input

```
37
```

Output

```
111;222;333;444;555;666;777;888;999;111111;222222;333333;444444;555555;666666
;777777;888888;999999;111111111;222222222;333333333;444444444;555555555;66666
6666;777777777;888888888;999999999
```

Example 2

Input

```
6
```

Output

```
6;66;222;444;666;888;6666;66666;222222;444444;666666;888888;6666666;66666666;
222222222;444444444;666666666;888888888
```

Solutions

Python

```
def main():
    modulo = int(input())
    result = []
    for n in range(1, 10):
        testing_number = n
        while testing_number < 1_000_000_000:
            if testing_number % modulo == 0:
                result.append(testing_number)
            testing_number = testing_number*10 + n
    result.sort()
    result_str = []
    for r in result:
        result_str.append(str(r))
    print(";".join(result_str))
if __name__ == '__main__':
    main()
```

C++

```
#include <iostream>
#include <vector>
```

```
#include <algorithm>
#include <string>
int main() {
    int modulo;
    std::cin >> modulo;
    std::vector<int> result;
    for (int n = 1; n < 10; n++) {
        long long testing_number = n;  // use long long to avoid integer
overflow
        while (testing_number < 1e9) {
            if (testing_number % modulo == 0) {
                result.push_back(testing_number);
            }
            testing_number = testing_number * 10 + n;
        }
    }
    std::sort(result.begin(), result.end());
    std::vector<std::string> result_str;
    for (int r : result) {
        result_str.push_back(std::to_string(r));
    }
    std::cout << result_str[0];
    for (int i = 1; i < result_str.size(); i++) {
        std::cout << ';' << result_str[i];
    }
    std::cout << std::endl;
    return 0;
}
```

10 Caesar's Message

9 points

Introduction

Since the old Roman empire, people wanted to send and receive messages which only the emitter and the receptor were able to understood.

One of the firts encryption methods, legendary attributed to Julius Caesar, was the letter substitution. This method consist in changing each letter in the message for another one.

Obviously, the emitter and the receptor need to know the letter substitution method, or the message wouldn't be able to be decrypted.

Exercise

You are asked to write a program that create an encripted message from an entered string.

This is the substitution sequence:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890
ZLKSMXNAJSHDGCBFVETWYQUIOP7593068421
```

Spaces will be translated as they are, just spaces.

Input

Input will consist in a text message. The message will be translated to capital letters and then it will transformed to the encoded message.

Output

The output will be the transformed/encoded message.

Example 1

Input

```
En un lugar de la mancha de cuyo nombre no quiero acordarme
```

Output

```
MC YC DYNZE SM DZ GZCKAZ SM KYOB CBGLEM CB VYJMEB ZKBESZEGM
```

Example 2

Input

```
Gaudeamus igitur iuvenes dum sumus
```

Output

```
NZYSMZGYT JNJWYE JYQMCMT SYG TYGYT
```

Solutions

Python

```python
# Author: Adrian Fernandez Alvarez (adrian.fernandez.alvarez@hp.com)
# Encrypts a given text
def encrypt(text):
    abc = ' ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890'
    abc_substituted = ' ZLKSMXNAJSHDGCBFVETWYQUIOP7593068421'
    cipher = ""
    for char in text:
        index = abc.find(char)
        if index >= 0:
            cipher += abc_substituted[index]
    return cipher
if __name__ == "__main__":
    text = input()
    try:
        text = text.upper()
    except:
        print("Please, enter a valid input\n")
        exit()
    cipher = encrypt(text)
    print(cipher)
```

C++

```cpp
#include <iostream>
#include <string>
using namespace std;
/*
  ::   .:::::::::::.
 ,;;   ;;,`;;;```.;;;
,[[[,,,[[[ `]]nnn]]'
```

```
"$$$"""$$$  $$$""
 888    "88o 888o
 MMM     YMM  YMMMb
  .,-:::::     ...    :::::::-.  .,::::::      .::    .   .:::::::.
::::::::..   .:::::::.
,;;;'````'  .;;;;;;;;.  ;;,   `';,;;;;;''''      ';;,  ;;  ;;;' ;;`;;
;;;;``;;;; ;;;`    `
[[[        ,[[     \[[,`[[     [[ [[cccc        '[[, [[, [[' ,[[ '[[,
[[[,/[[[' '[==/[[[[,
`88bo,__,o,"888,_ _,88P 888_,o8P' 888oo,__        "88"888   888   888,888b
"88bo,88b    dP
  "YUMMMMMP" "YMMMMMP"  MMMMP"`   """"YUMMM         "M "M"   YMM   ""` MMMM
"W"  "YMmMY"

Encript a string
From the old Roman empire, people wants to send and receive messages which
only the emiiter and
the receptor were able to understood
One of the firts encryption methos was the letter substitution. This method
consist in change
each letter in the message for another one. Obviusly, the emmiter and the
receptor has to be
owner of the letter substitution method, or the message should be a mess for
enyone
This exercice wants to create an encription message starting for a string
entered
This is the substitution sequence
ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890
ZLKSMXNAJSHDGCBFVETWYQUIOP7593068421
*/
string encrypt( string cString )
{
    // Convert string to uppercase
    string cUpper = "";
    for( int x=0; x < cString.length(); x++ )
```

```
      cUpper += toupper( cString[x] );
    cString = cUpper;
    // Encrypt
    string cOriginal = " ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890";
    string cCambio   = " ZLKSMXNAJSHDGCBFVETWYQUIOP7593068421";
    string cEncriptado = "";
    int n = 1;
    for( int n=0; n<cString.length(); n++ )
    {
      for( int x=0; x<cOriginal.length(); x++ )
      {
        if( cOriginal.substr( x , 1 ) == cString.substr( n , 1 ) )
        {
          cEncriptado += cCambio.substr( x , 1 );
          break;
        }
      }
    }
  return cEncriptado;
}
int main()
{
    string cCadena;
    getline( cin , cCadena );
    // Show encrypted string
    cout << encrypt( cCadena );
}
```

11 Escape speed calculator

9 points

Introduction

In physics escape velocity is the minimum speed needed for a free, non-propelled object (a rocket) to escape from the gravitational influence of a massive body (that is, a planet or star). Escape velocity rises with the planet's mass and falls with the escaping object's distance from its center.

The escape velocity for a body(planet or star), is calculated by the formula:

```
v = sqrt((2*G*M)/r))
```

where G is the universal gravitational constant (G = 6.67×10−11 m3·kg−1·s−2), M the mass of the body to be escaped from, and r the distance from the center of mass of the body to the object (if the rocket is in the surface, this distance will the radius of the planet or star).

The escape velocity from Earth's surface is about 11186 m/s because Earth's mass is 6.0 x 10^24 kilogram and it's radius is 6.4 x 10^6 meters. Therefore:

```
v = sqrt(2*6.67x10^-11*6.0*10^24/6.4*10^6) = sqrt(13.33*6.0*10^13/6.4*10^6) =
sqrt (1.2496875*10^8) = 11186 m/s.
```

Exercise

You are asked to write a Escape velocity calculator for a rocket that's on the surface of a planet. The program will receive the mass of the planet in kilograms and the radius of the planet in meters.

The calculator will return the escape velocity in m/s.

Input

The input will be three lines:

The first line will the planet's name.

The second line will be the mass of the planet in kilograms.

The third line will be radius of the planet in meters.

The second and third input lines will have the following structure:

```
<number> x 10^<exponent>
```

In this way, 6.4 x 10^6 will mean 6 400 000.

Output

The output will be a single line indicating the of the escape velocity in m/s (meters per second). It must be an integer value (no decimal numbers) rounded to the closest integer value.

Example 1

Input

```
Tierra
6.0 x 10^24
6.4 x 10^6
```

Output

```
11186
```

Example 2

Input

```
Moon
0.07346 x 10^24
1.738 x 10^6
```

Output

```
2380
```

Solutions

Python

```
import sys
import re
import math
if __name__ == "__main__":
    G = float(6.67 * pow(10, -11))
    planet = input("")
    mass_str = input("")
    values = mass_str.split("x")
    exps = values[1].split("^")
    valor = float(values[0])
    exp = float(exps[1])
    mass = valor*pow(10, exp)
    radius_str = input("")
    values = radius_str.split("x")
    exps = values[1].split("^")
    valor = float(values[0])
    exp = float(exps[1])
    radius = valor*pow(10, exp)
    vel = round(math.sqrt(2 * G * mass / radius))
    print(vel)
```

C++

```
#include <stdio.h>
#include <vector>
#include <string>
#include <cmath>
#include <iostream>
int main()
{
    std::vector<std::pair<std::string, float>> listado;
    std::string delimiter="^";
    std::string planet="";
    std::string value="";
    std::string por="";
    std::string exponent="";
    float G=6.67 * pow(10,-11);
    //Read planet name
    getline(std::cin, planet);
    //Read mass (As there are spaces in the input, we can do it with cin)
    std::cin >> value;
    std::cin >> por;
    std::cin >> exponent;
    if (exponent.substr(0, exponent.find(delimiter))!="10")
    {
        std::cout << "Error";
        return -1;
    }
    float valor=std::stof(value);
    int exp=std::stoi(exponent.substr(exponent.find(delimiter)+1,
exponent.size()));
    double mass=valor*pow(10,exp);
    //Read radius
    std::cin >> value;
    std::cin >> por;
    std::cin >> exponent;
    if (exponent.substr(0, exponent.find(delimiter))!="10")
    {
        std::cout << "Error";
        return -1;
    }
    valor=std::stof(value);
    exp=std::stoi(exponent.substr(exponent.find(delimiter)+1,
exponent.size()));
    long radius=valor*pow(10,exp);
    //Calculate escape speed
    int vel=(int)round(sqrt(2 * G * mass / radius));
    std::cout << vel << std::endl;
    return 0;
}
```

12 The Cult of the Curly Numbers

9 points

Introduction

A new thinking movement considers that only the numbers with curly traces are worthy of God. Only the digits 0, 3, 6, 8 and 9 are considered as curly, because all the other have some straight parts and that is considered to be a "dirty" number. As they don't know exactly how many valid numbers they have, they have asked you to develop a program that helps us identify, for a given number of digits, how many positive integers (including zero) with those digits are formed only by curly traces and thus are worthy numbers.

Exercise

The program will receive a positive integer indicating the number of desired digits and will return the number of all the posible numbers with those digits than can be created using only curly numbers.

Please notice that a number starting by "0" has a smaller number of digits. For example, the number 0986 counts as having 3 digits and not four, because the zero on the left doesn't count. However, the number 0 itself counts as a one digit number.

Input

A positive number (number of digits for the numbers to be counted).

Output

Number of numbers with those digits that are only formed with curly digits.

Example 1

Input

```
1
```

Output

```
5
```

Example 2

Input

2

Output

20

For this last example, the numbers found are 30, 33, 36... up to 99, which is the last 2 digit number with valid curly digits.

Solutions

Python

```
CURLY_NUMBERS = {'0', '3', '6', '8', '9'}
def main():
    n = int(input())
    min = pow(10, n-1) if n > 1 else 0
    max = pow(10, n)
    only_curly = 0
    for i in range(min, max):
        number_str = str(i)
        is_curly = True
        for j in range(0, len(number_str)):
            if not number_str[j] in CURLY_NUMBERS:
                is_curly = False
                break
        if is_curly:
            only_curly += 1
    print(only_curly)
if __name__ == "__main__":
    main()
```

C++

```
#include <iostream>
#include <math.h>
#include <string>
int main()
{
    int n;
    std::cin >> n;
    int min = n > 1 ? pow(10, n - 1) : 0;
    int max = pow(10, n);
    int onlyCurly = 0;
    for (int i = min; i < max; i++)
    {
        std::string numberStr = std::to_string(i);
        bool isCurly = true;
        for (int j = 0; j < numberStr.length(); j++)
        {
            char numberChar = numberStr[j];
            if (numberChar != '0' && numberChar != '3' && numberChar != '6'
&& numberChar != '8' && numberChar != '9')
            {
                isCurly = false;
                break;
            }
        }
        if (isCurly)
        {
            onlyCurly++;
        }
    }
    std::cout << onlyCurly << std::endl;
}
```

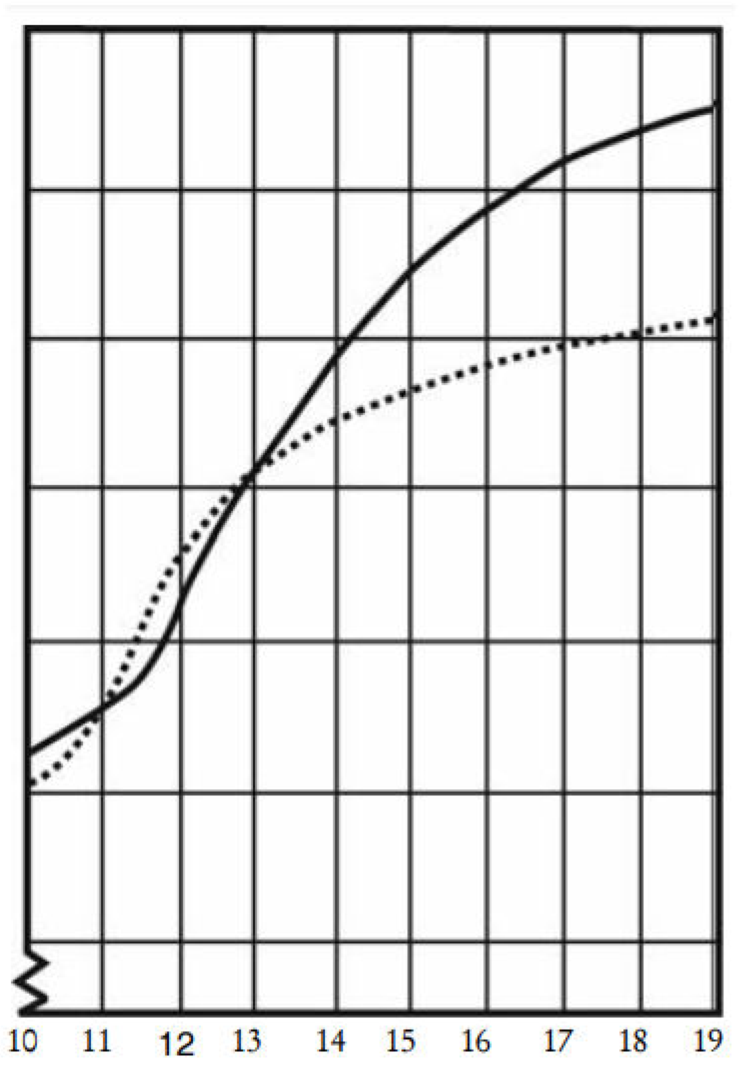
13 The growth curve

9 points

Introduction

At the Ministry of Education they want to know how is the Spanish male growth curve between ages 10 to 19 compared with the same curve in the Netherlands. Unfortunately, during the transference of the plot from the Netherlands, the measure values have been lost.

The growth curve in the Netherlands is the following (only the solid line):



This curve measures the average height of Dutch boys between ages 10 and 19.... but... what do the horizontal lines mean?. The Spanish ministry does not certainly know. Fortunately, the Dutch authorities also sent the polynomial function representing the same curve. This polynomial function is the following:

$$\frac{x^9}{12096} - \frac{29x^8}{2688} + \frac{349x^7}{560} - \frac{60233x^6}{2880} + \frac{1294009x^5}{2880} - \frac{7381433x^4}{1152} + \frac{1833933389x^3}{30240} - \frac{3702218497x^2}{10080} + \frac{1084584583x}{840} - 2006804$$

```
f(x) = x^9/12096 - 29*x^8/2688 + 349*x^7/560 - 60233*x^6/2880 +
1294009*x^5/2880 - 7381433*x^4/1152 + 1833933389*x^3/30240 -
3702218497*x^2/10080 + 1084584583*x/840 - 2006804
```

This is actually great because now Spanish authorities can know the height of Dutch boys aged 13.5, 14.5, etc.

You have been asked to write a program decoding the polynomial function, so that Spanish authorities can see which are the average male heights in the Netherlands at different ages.

Exercise

The program will receive a value of the age and will return the average height from the polynomial function, with two decimal digits.

Input

The program will receive a value of the age and will return the average height from the polynomial function. The input age can be a value from 10 to 19, both integer or non integer (13.5, 13.4...).

Output

The output will be a single value indicating the average height at that age with two decimal values.

If value is under 10 or over 19, the program will return:

```
ERROR
```

Example 1

Input

```
13
```

Output

```
162.00
```

Example 2

Input

```
13.5
```

Output

```
165.79
```

Example 3

Input

```
3.5
```

Output

```
ERROR
```

Solutions

Python

```
def growth_poly(age: float) -> None:
    # f(x) = 1.0//12096*x^9 - 29/2688*x^8 + 349/560*x^7 - 60233/2880*x^6 +
1294009/2880*x^5 - 7381433/1152*x^4 + 1833933389/30240*x^3 -
3702218497/10080*x^2 + 1084584583/840*x - 2006804
    if age < 10.0 or age > 19.0:
        print("ERROR")
    else:
        coeff = [1.0/12096, -29/2688, 349/560, -60233/2880, 1294009/2880, -
7381433 /
                 1152, 1833933389/30240, -3702218497/10080, 1084584583/840, -
2006804]
        x = age
        result = coeff[0]*x**9 + coeff[1]*x**8 + coeff[2]*x**7 +
coeff[3]*x**6 + coeff[4] * \
            x**5 + coeff[5]*x**4 + coeff[6]*x**3 + \
            coeff[7]*x**2 + coeff[8]*x + coeff[9]
        print(f"{result:5.2f}")
def main():
    age = float(input())
    growth_poly(age)
```

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

C++

```
#include <string>
#include <iostream>
#include <vector>
#include <sstream>
#include <iomanip>
float funcion(float X)
{
      //f(x) = x^9/12096 - 29*x^8/2688 + 349*x^7/560 - 60233*x^6/2880 +
1294009*x^5/2880 - 7381433*x^4/1152 + 1833933389*x^3/30240 -
3702218497*x^2/10080 + 1084584583*x/840 - 2006804
      double equis=X;
      //std::cout << "\nX es " << equis;
      double last=(1084584583.0/840.0)*equis;
      //std::cout << "\nterm es " << std::fixed << last;
      equis=equis*X; //X2
      //std::cout << "\nX2 es " << equis;
      double oct=(3702218497.0/10080.0)*equis;
      //std::cout << "\nterm es " << oct;
      equis=equis*X;  //X3
      //std::cout << "\nX3 es " << equis;
      double sept=(1833933389.0/30240.0)*equis;
      //std::cout << "\nterm es " << sept;
      equis=equis*X;  //X4
      //std::cout << "\nX4 es " << equis;
      double sexto=(7381433.0/1152.0)*equis;
      //std::cout << "\nterm es " << sexto;
      equis=equis*X;  //X5
      //std::cout << "\nX5 es " << equis;
      double quinto=(1294009.0/2880.0)*equis;
      //std::cout << "\nterm es " << quinto;
      equis=equis*X;  //X6
      //std::cout << "\nX6 es " << equis;
      double cuarto=(60233.0/2880.0)*equis;
      //std::cout << "\nterm es " << cuarto;
      equis=equis*X;  //X7
      //std::cout << "\nX7 es " << equis;
      double ter=(349.0/560.0)*equis;
      //std::cout << "\nterm es " << ter;
      equis=equis*X;  //X8
      //std::cout << "\nX8 es " << equis;
      double sec =(29.0/2688.0)*equis;
      //std::cout << "\nterm es " << sec;
      equis=equis*X;  //X9
      //std::cout << "\nX9 es " << equis;
      double prim=equis/12096.0;
      //std::cout << "\nterm es " << prim;
      float ret=(float)(prim - sec + ter -cuarto + quinto -sexto + sept -oct
+ last - 2006804.0);
      //std::cout << "\nEL RESULTADO FINAL ES " << ret;
      return ret;
}
int main(int argc, char **argv)
{
    std::string input_text="";
    //INPUT
    std::cin >> input_text;
    float input_val=std::stof(input_text);
    if (input_val<10.0 || input_val>19.0)
        std::cout<<"ERROR";
    else
        std::cout<< std::setprecision(2) << std::fixed<<funcion(input_val);
    return 0;
}
```

14 Communicating with ETs

10 points

Introduction

In the SETI (Search for ExtraTerrestrial intelligence) project they have found that a remote star is sending radio pulses in two different frequencies. These pulses turn out to be different length binary numbers sent simultaneously using different frequencies and with a silence period at the end of each number. There's been quickly a conclussion which is in all TVs: there are intelligent ETs!!!

The odd thing scientists are wondering is why these binary numbers are being received through two different frequencies. Fortunately, a prodigious mind has found the reason: the intelligent ETs are testing us, they want us to return the sum of the digits received to show our intelligence... but in the SETI projectiscientists don't know how to sum up binary numbers (lately their interviewing process has gotten too lax)...

Can you help them?

Exercise

You are asked to write a program to sum up two binary numbers. Remember that a binary number is a number represented only with 0 and 1.

Input

Input will consist of two lines. The first line will be the first binary number, and the second line will be the second binary number. Both numbers can be of different length.

Output

The output value will be sum of the binary numbers.

Example 1

Input

```
1010
11111111
```

Output

```
100001001
```

Example 2

Input

```
1000111
10011
```

Output

```
1011010
```

Solutions

Python

```python
def binary_sum(a, b):
    return bin(int(a, 2) + int(b, 2))[2:]
def main():
    a = input()
    b = input()
    result = binary_sum(a, b)
    print(result)
if __name__ == '__main__':
    main()
```

C++

```cpp
#include <iostream>
#include <string>
using namespace std;
/*
  ::   .:::::::::::.
 ,;;   ;;,`;;;```.;;;
,[[[,,,[[[ `]]nnn]]'
"$$$"""$$$  $$$""
 888   "88o 888o
 MMM    YMM YMMMb
  .,-:::::     ...    :::::::-.  .,::::::      .::    .   .::::::.
:::::::..   .::::::.
,;;;'````'  .;;;;;;;.  ;;,   `';,;;;;''''      ';;,  ;;  ;;;' ;;`;;
;;;;``;;;; ;;;`    `
```

```
[[[        ,[[     \[[,`[[     [[ [[cccc        '[[, [[, [[' ,[[ '[[,
[[[,/[[[' '[==/[[[[,
`88bo,__,o,"888,_ _,88P 888_,o8P' 888oo,__         "88"888   888   888,888b
"88bo,88b    dP
  "YUMMMMMP" "YMMMMMP"  MMMMP"`   """"YUMMM         "M "M"   YMM   ""` MMMM
"W"  "YMmMY"

Sum two binary numbers
A binary number is a number represented only with 0 and 1.
*/
string binary_add( string cBin1, string cBin2 ) {
    int size = max( cBin1.size() , cBin2.size() );
    int temp = 0;
    string result_str = "";
    for( auto i = 0; i < size; ++i )
    {
        int digit1 = ( i + 1 <= cBin1.size() ) ? cBin1[cBin1.size() - 1 - i]
- '0' : 0;
        int digit2 = ( i + 1 <= cBin2.size() ) ? cBin2[cBin2.size() - 1 - i]
- '0' : 0;
        int number = ( digit1 + digit2 + temp);
        temp = number / 2;
        result_str = to_string( number % 2 ) + result_str;
    }
    if( temp > 0 )
    {
        result_str = to_string( temp ) + result_str;
    }
    return result_str;
}
int main()
{
    string  cBin1 , cBin2;
    cin >> cBin1;
    cin >> cBin2;
    //
    cout << binary_add( cBin1 , cBin2 );
}
```

15 Stop the virus

10 points

Introduction

Scientists have discovered that a very dangerous virus generates holes in the lungs following a Fibonacci series. The size of the hole is 1/10 of the Fibonacci sequence value.

The Fibonacci sequence is a series of numbers where each number is the sum of the two preceeding numbers (except the two first, which are 0 and 1).

Therefore the Fibonacci series is 0,1,1,2,3,5,8,13... "ad infinitum".

On first day of attack, nothing happens, but starting on the second day of attack, a hole of size 0.1mm2 appear. On fourth day of attack, the hole is 0.2mm2, on 7th day of infection, the hole has grown to 0.8mm2, etc.

Fortunately a medicine has been discovered, and curious enough, it works following a Fibonacci sequence as well. Once the medicine is administered, the lung holes stop growing and start decreasing following a Fibonacci sequence: the first day they do not decrease at all (but they don't grow either), the second and third day the have decreased 0.1mm2, the third day taking the pills they have decreased 0.2mm2, etc.

Scientists have also discovered that if the medicine is administered after day 17th (day 17th is the last allowed option), the patient dies.

Exercise

You are asked to write a program to calculate how big are those holes, given the infection day and the medicine administration day. Let's imagine an example where the input values are:

```
9 6
```

This means that the infection is in it's 9th day and the medicine started being treated in the 6th day. Fibonacci is as follows:

```
Day:       1 2 3 4 5 6 7 8  9
Fibonacci: 0 1 1 2 3 5 8 13 21
Infection: 0 1 1 2 3 5 5 4  4  <-- this marks the size of the holes
Cure:                  0 1  1
```

Therefore, from day after administration, the infection stops growing and starts decreasing following a Fibonacci sequence.

In day 9th, the lung hole is 0.4 mm2.

Input

The program will receive a line with two positive numbers (Greater than zero) indicating the day of infection and the day of medicine administration.

The second number MUST be lower than the first one.

It's also possible to introduce only one number indicating that the medicine has not been administered yet.

Output

The output will be the size of the holes given the day of infection and the day of administration of the medicine (if any). The format of the output will be like:

```
0.1 mm2
```

With one decimal place.

If the holes are zero (can't be lower than zero) the output will be:

```
The patient is cured!!!
```

If the medicine was administered after the 17th day, the output will be:

```
Unfortunately, the patient died...
```

Example 1

Input

```
9 6
```

Output

```
0.4 mm2
```

Example 2

Input

```
28 8
```

Output

```
Unfortunately, the patient died...
```

Example 3

Input

```
8 7
```

Output

```
0.8 mm2
```

Solutions

Python

```python
def Fib(n):
    if n==1:
        return 0
    elif n==2:
        return 1
    else:
        return Fib(n-1)+Fib(n-2)
inn=input("")
infection_day = int(inn.split(' ')[0])
cure_day = int(inn.split(' ')[1])
if (infection_day - cure_day)>17:
    print("Unfortunately, the patient died...")
else:
    infection_spread=Fib(cure_day)
    cure_spread=Fib(infection_day - cure_day)
    if (infection_spread - cure_spread)<=0:
        print("The patient is cured!!!")
    else:
        print(0.1*(infection_spread - cure_spread), "mm2");
```

C++

```cpp
#include <stdio.h>
#include <vector>
#include <string>
#include <cmath>
```

```
#include <iostream>
#include <iomanip>
int fibonacci(int i)
{
    if (i == 1)
        return 0;
    if (i == 2)
        return 1;
    return fibonacci(i - 1) + fibonacci(i - 2);
}
int main()
{
    std::string infection_str = "";
    std::string cure_str = "";
    std::cin >> infection_str;
    std::cin >> cure_str;
    int infection_day = std::stoi(infection_str);
    int cure_day = std::stoi(cure_str);
    if ((infection_day - cure_day) > 17)
    {
        std::cout << "Unfortunately, the patient died...";
        return 0;
    }
    int infection_spread = fibonacci(cure_day);
    int cure_spread = fibonacci(infection_day - cure_day);
    if ((infection_spread - cure_spread) <= 0)
    {
        std::cout << "The patient is cured!!!";
    }
    else
    {
        std::cout << std::fixed << ::std::setprecision(1) << 0.1 *
(infection_spread - cure_spread) << " mm2";
    }
    return 0;
    }
```

16 Divisible palindromes

11 points

Introduction

A palindrome is a number that can be read starting from the front and from the tail and the read number will be same. For instance number 18481 can be read starting from the beginning or from the end. It's a palindrome.

Exercise

You are asked to write a program that finds the first N palindromes (starting from 10), which are multiples of M, being N and M two values indicated as input.

For example, and input line like:

```
3 5
```

will return:

```
33 66 99 111 141
```

As these are the first 5 palindrome numbers mutiples of 3.

Input

The input will consist in a line with two numbers:

```
<M> <N>
```

Being *M* the divisor and *N* the amount of numbers to be shown.

Output

The output will be the requested palindrome numbers separated by an space.

Example 1

Input

```
3 5
```

Output

```
33 66 99 111 141
```

Example 2

Input

```
109 3
```

Output

```
545 5995 15151
```

Solutions

Python

```python
def is_palindrome(number):
    number_str = str(number)
    # a number is a palindrome if it equals it in reverse
    return number_str == number_str[::-1]
def main():
    multiples_of, n_palindromes = (int(v) for v in input().split())
    # start at the multiples of value which is 10 or above
    if multiples_of >= 10:
        current = multiples_of
    else:
        current = multiples_of
        while current < 10:
            current += multiples_of
    palindromes_found = []
    while len(palindromes_found) < n_palindromes:
        if (is_palindrome(current)):
            palindromes_found.append(current)
        current += multiples_of
    print(*palindromes_found, sep=' ')  # print the list separated by spaces
if __name__ == "__main__":
    main()
```

C++

```cpp
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>
bool isPalindrome(int number)
{
    std::string numberStr = std::to_string(number);
    std::string reversed(numberStr);
    std::reverse(reversed.begin(), reversed.end());
    // a number is a palindrome if it equals it in reverse
    return numberStr == reversed;
}
int main()
{
    int multiplesOf;
    int nPalindromes;
    std::cin >> multiplesOf;
    std::cin >> nPalindromes;
    // start at the multiples of value which is 10 or above
    int current;
    if (multiplesOf >= 10)
        current = multiplesOf;
    else
    {
        current = multiplesOf;
        while (current < 10)
            current += multiplesOf;
    }
    std::vector<int> palindromesFound;
    while (palindromesFound.size() <
static_cast<std::vector<int>::size_type>(nPalindromes))
    {
        if (isPalindrome(current))
            palindromesFound.push_back(current);
        current += multiplesOf;
    }
    // print the list separated by spaces
    for (int i = 0; i < nPalindromes; i++)
    {
        std::cout << palindromesFound[i];
        if (i != nPalindromes - 1) // separator for all but the last one
```

```
            std::cout << " ";
        }
        std::cout << std::endl;
    }
```

17 The Dancing Numbers

11 points

Introduction

For numbers to be considered dancing numbers, three numbers composed by three digits are required, but permutated in the order in which the digit in the first position shifts to the last. In fact, these are the numbers *abc*, *bca* (*a* moves to the back), and *cab* (now *b* moves to the back).

Exercise

Write a program that for a sufficiently large input number calculates the value of the digits *a*, *b*, and *c* so that the input number is equal to the multiplication of the three permutated numbers composed by these digits.

Input

An integer number *n*.

Output

In case the three numbers are found:

```
abc * bca * cab = <n>
```

Otherwise, it should print the message:

```
There are no dancing numbers for <n>.
```

Where <*n*> is the input number.

Dancing numbers have to be printed in descending order, starting by the larguest of them.

Example 1

Input

```
328245326
```

Output

```
983 * 839 * 398 = 328245326
```

Example 2

Input

```
2000023
```

Output

```
There are no dancing numbers for 2000023.
```

Solutions

Python

```python
def read_number() -> int:
    return int(input())
def permutate_digit(number: int) -> int:
    number = str(number)
    assert len(number) == 3, f"The number of digits must be three. Got:
{len(number)}."
    number = number[1:3] + number[0]
    return int(number)
def get_main_dancers() -> list:
    main_dancers = []
    for i in range(1, 10):
        main_dancers.append(int(f'{i}'*3))
        for j in range(i+1, 10):
            main_dancers.append(int(f'{i}'*2 + f'{j}'))
            main_dancers.append(int(f'{j}'*2 + f'{i}'))
            for k in range(j+1, 10):
                main_dancers.append(int(f'{i}{j}{k}'))
                main_dancers.append(int(f'{i}{k}{j}'))
    return main_dancers
def find_dancers(number: int) -> list:
    for d1 in get_main_dancers():
        if number % d1 == 0:
            d2 = permutate_digit(d1)
            d3 = permutate_digit(d2)
            if d1 * d2 * d3 == number:
                return [d1, d2, d3]
    return []
def main() -> None:
    number = read_number()
    dancing_numbers = find_dancers(number)
    # sort dancing numbers from largest to smallest
    dancing_numbers.sort(reverse=True)
    if len(dancing_numbers) == 0:
        print(f"There are no dancing numbers for {number}.")
    else:
        print(f"{dancing_numbers[0]} * {dancing_numbers[1]} *
{dancing_numbers[2]} = {number}")
if __name__ == '__main__':
    main()
```

C++

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <cassert>
#include <string>
using namespace std;
int read_number()
{
    int num;
    cin >> num;
    return num;
}
int permutate_digit(int number)
{
    string str = to_string(number);
    str = str.substr(1, 2) + str[0];
    return stoi(str);
}
vector<int> get_main_dancers()
{
    vector<int> main_dancers;
    for (int i = 1; i < 10; i++)
    {
        main_dancers.push_back(i * 111);
        for (int j = i + 1; j < 10; j++)
        {
            main_dancers.push_back(i * 11 + j * 10);
            main_dancers.push_back(j * 11 + i * 10);
            for (int k = j + 1; k < 10; k++)
            {
                main_dancers.push_back(i * 100 + j * 10 + k * 1);
                main_dancers.push_back(i * 100 + k * 10 + j * 1);
            }
        }
    }
    return main_dancers;
}
vector<int> find_dancers(int number)
{
    vector<int> main_dancers = get_main_dancers();
    for (auto d1 : main_dancers)
    {
        if (number % d1 == 0)
        {
            int d2 = permutate_digit(d1);
            int d3 = permutate_digit(d2);
            if (d1 * d2 * d3 == number)
            {
                return {d1, d2, d3};
            }
        }
    }
    return {};
}
int main()
{
    int number = read_number();
    vector<int> dancing_numbers = find_dancers(number);
    sort(dancing_numbers.rbegin(), dancing_numbers.rend());
    if (dancing_numbers.empty())
    {
        cout << "There are no dancing numbers for " << number << "." << endl;
    }
    else
    {
        cout << dancing_numbers[0] << " * " << dancing_numbers[1] << " * " << 
dancing_numbers[2] << " = " << number << endl;
    }
    return 0;
}
```

18 Kaprekar digit sum

12 points

Introduction

The Kaprekar mathematician D. R. Kaprekar (1905-1986) made some amazing findings about digit sum.

An example of his findings: choose a positive number and sum it with the sum of its digits. For instance, with number 23, add to it 2 and 3, so 23+2+3=28. Starting again with 28, we could write a series of always-increasing numbers (28+2+8=38, 38+3+8=49...).

Curiously enough, it's BEEN IMPOSSIBLE to find a formula to determine the nth position of this series, starting with a given number: you need to follow the series step by step.

Exercise

Code a program that given an initial number an a position in the series, obtain the number in that position in the series.

Input

The input will be two lines.

The first line is the first number of the series.

The second number is the number of elements the series will have after the initial number.

Both numbers will be positive, non-zero, integer numbers. The first number will be no greater than 1 million. The second number will be no greater than 1000.

Output

The output will be a single number with the last number of series.

If any of the constraints previously indicated is not complied, the output will be:

text

INVALID NUMBER

Example 1

Input

23

11

Output

115

```
The explanation is:
23+5=28 (n=1); 28+10=38 (n=2); 38+11=49 (n=3); 49+13=62 (n=4); 62+8=70 (n=5);
70+7=77 (n=6); 77+14=91 (n=7); 91+10=101 (n=8); 101+2=103 (n=9); 103+4=107
(n=10); 107+8=115 (n=11)
```

Example 2

Input

1234

332

Output

6655

Solutions

Python

```
import sys

initial = input().strip()

iter = input().strip()

if len(initial) > 6 or not initial:

    print("INVALID NUMBER")

    sys.exit()

if initial.isalpha():

    print("INVALID NUMBER")

    sys.exit()
```

```
if len(iter) > 3 or not initial:
    print("INVALID NUMBER")
    sys.exit()
if iter.isalpha():
    print("INVALID NUMBER")
    sys.exit()
suma = int(initial)
for x in range(int(iter)):
    l = len(str(suma))
    aux = 0
    for i in range(l):
        numero = str(suma)
        aux += int(numero[i])
        if i == l-1:
            suma += aux
print(suma)
```

C++

```
#include <string>
#include <iostream>
int main(int argc, char argv)
{
    std::string initial;
```

```
std::string iter;

long num = 0;

long count = 0;

std::cin >> initial;

std::cin >> iter;

if ((initial.length() > 6) || initial.empty())

{

    std::cout << "INVALID NUMBER";

    return 0;

}

if (initial.find_first_not_of("0123456789") != std::string::npos)

{

    std::cout << "INVALID NUMBER";

    return 0;

}

if ((iter.length() > 3) || initial.empty())

{

    std::cout << "INVALID NUMBER";

    return 0;

}

if (iter.find_first_not_of("0123456789") != std::string::npos)

{
        std::cout << "INVALID NUMBER";
        return 0;
    }
    num = stol(initial);
    count = stol(iter);
    std::string temp = initial;
    for (unsigned long i = 0; i < count; i++)
    {
        for (int pos = 0; pos < temp.length(); pos++)
        {
            char a = temp[pos] - '0';
            num = num + a;
        }
        temp = std::to_string(num);
    }
    std::cout << num << std::endl;
    return 0;
}
```

19 The McCarthy 91 function

12 points

Introduction

The McCarthy 91 function is defined as follows:

```
       |-> n - 10 (if n > 100)
M91(n)=|
       |-> M91(M91(n+11)) (if n <= 100)
```

This function is special because all numbers under 100 will end up in 91.

Generalizing it, the McCarthy function will be:

```
            |-> n - XX (if n > ZZ)
McCarthy(n)=|
            |-> McCarthy(McCarthy(n+YY)) (if n <= ZZ)
```

Exercise

You are asked to write a program to calculate the McCarthy function of a given value.

Input

The first input line will consist in three numeric positive values XX, YY and ZZ, separated by a comma:

```
XX,YY,ZZ
```

The second input line will consist in a numeric positive number not greater than 9999, which is the value of *n*.

Output

The program will return McCarthy function of the given integer.

Example 1

Input

```
10,11,100
74
```

Output

```
91
```

Example 2

Input

```
10,11,100
774
```

Output

```
764
```

Example 3

Input

```
10,20,299
45
```

Output

```
295
```

Solutions

Python

```
import sys
XX = None
YY = None
ZZ = None
def mc_carthy(number):
    if (number > ZZ):
        return (number - XX)
    else:
        return mc_carthy(mc_carthy(number+YY))
if __name__ == "__main__":
    args = input("").split(',')
    XX = int(args[0])
    YY = int(args[1])
    ZZ = int(args[2])
    number = int(input())
    print(mc_carthy(number))
```

C++

```
#include <string>
#include <iostream>
#include <vector>
#include <sstream>
int XX, YY, ZZ;
int mccarthy(int n)
{
      if (n>ZZ)
            return n-XX;
      else
            return mccarthy(mccarthy(n+YY));
}
int main(int argc, char **argv)
{
    std::vector<int> vect;
    std::string input_text="";
    std::string input_text2="";
    std::cin >> input_text;
    std::stringstream ss(input_text);
    for (int i; ss >> i;) {
        vect.push_back(i);
        if (ss.peek() == ',')
            ss.ignore();
    }
    XX=vect[0];
    YY=vect[1];
    ZZ=vect[2];
    std::cin >> input_text2;
    int input_val=std::stoi(input_text2);

    std::cout<<mccarthy(input_val);
    return 0;
}
```

20 Where's the lava?

12 points

Introduction

In a Canarian island there's been a volcano eruption. As it's well known, lava destroys everything in front of it as it moves forward. The island's Cabildo has asked you to develop a program to allow residents to know how much time they have to empty their houses before the lava destroys them.

Exercise

As lava moves forward at a variable speed, the program will receive two parameters: the distance of the user's house to the front of lava and the speed that, in this moment, has the front of lava.

The first parameter will be the distance IN METERS, and the second parameter (second input line) will be the lava speed IN CENTIMETERS PER HOUR. The program will return the total time the user has left to empty their house and it will be returned with following format:

```
XX days,YY hours,ZZ minutes,VV seconds.
```

If XX, YY, ZZ or VV are zero, they will not be included. Seconds will be rounded to the closest integer value.

Input

The program will receive two parameter in two different lines. The first line will be the distance IN METERS, and the second parameter (second input line) will be the speed IN CENTIMETERS PER HOUR.

BOTH WILL BE integer values (if a fractional number is introduced, i.e. 90.87, only the integer part will be taken into account, 90 in this case). If any negative number is introduced, the program will not show anything. However the tests for this problem will only use integer pòsitive numbers.

Output

The program will return the total time the user has to empty their house and it will be returned with following format:

```
XX days,YY hours,ZZ minutes,VV seconds.
```

If XX, YY, ZZ or VV are zero, that time measure will not be mentioned. Please notice that there are no spaces surrounding the commas.

Seconds will be rounded to the most aproximate integer (i.e 3.8 seconds will be shown as 4 seconds).

Example 1

Input

```
583
564
```

Output

```
04 days,07 hours,22 minutes,08 seconds.
```

Example 2

Input

```
123
123
```

Output

```
04 days,04 hours.
```

Solutions

Python

```
def main():
    distance_m = int(input())
    speed_cm_h = int(input())
    speed_m_s = (speed_cm_h / 100) / 3600
    time_s = distance_m/speed_m_s
    days = int(time_s // 86400)
    remaining_s = ((time_s / 86400) - days) * 86400
    hours = int(remaining_s // 3600)
    remaining_s = ((remaining_s / 3600) - hours) * 3600
    minutes = int(remaining_s // 60)
    remaining_s = ((remaining_s / 60) - minutes) * 60
    seconds = int(round(remaining_s))
    if seconds == 60:  # if rounding up has gone to 60, add a minute and
remove seconds
        seconds = 0
```

```python
            minutes += 1
        if days > 0:
            print(f"{days:02d} days", end='')
        if hours > 0:
            if (days > 0):
                print(",", end='')
            print(f"{hours:02d} hours", end='')
        if minutes > 0:
            if (days > 0 or hours > 0):
                print(",", end='')
            print(f"{minutes:02d} minutes", end='')
        if seconds > 0:
            if (days > 0 or hours > 0 or minutes > 0):
                print(",", end='')
            print(f"{seconds:02d} seconds", end='')
        print(".")
if __name__ == "__main__":
    main()
```

C++

```cpp
#include <string>
#include <iostream>
#include <algorithm>
#include <math.h>
#include <iomanip>
int main(int argc, char **argv)
{
    std::string meters_str="";
    std::string speed_str="";
    std::cin >> meters_str;
    std::cin >> speed_str;
    int centimeters=std::stoi(meters_str)*100;
    int speed=std::stoi(speed_str);
    int days=0;
```

```
double time_init=((1.0)*centimeters)/((1.0)*speed);  //in hours
//days
if (time_init>=24)
{
    days=time_init/24.0;
  time_init=time_init - (24*days);
}
//hours
int hours=(int)time_init;
time_init=time_init-hours; //
//minutes
time_init=time_init*60;
int minutes=(int)time_init;
time_init=time_init-minutes; //t
//seconds
time_init=time_init*60;
int seconds=(int)round(time_init);
std::stringstream ss;
if (days>0)
    ss << std::setfill('0') << std::setw(2) << days << " days,";
if (hours>0)
    ss << std::setfill('0') << std::setw(2) << hours << " hours,";
if (minutes>0)
    ss << std::setfill('0') << std::setw(2) << minutes << " minutes,";
if (seconds>0)
    ss << std::setfill('0') << std::setw(2) << seconds << " seconds";
char last_char;
ss.seekg(-1,ss.end);//get to the last character in the buffer
ss>>last_char;
std::string s=ss.str();
if (last_char==',')
{
    std::replace(s.end()-1, s.end(), ',', '.');
    std::cout << s;
}
```

```
    else
    {
        ss.seekg (0, ss.end);
        ss<< ".";
        std::cout << ss.str();
    }
    return 0;
}
```

21 Check my password

13 points

Introduction

In HP weak passwords are not allowed. Future employees need to know if a password is robust or not. To do so they are asked to develop a program to check for weak passwords. Do you want to be an HP employee?

Exercise

Different HP centers may require different robustness in their employees' passwords. For this reason the program will require two parameters, the first one will the HP center password template and the second will be the employee password.

The program will simply indicate if the password is robust or not depending on this two input parameters.

Input

The input line will consist in two parameters:

`<template> <password>`

separated by a blank space.

The template is conformed by the following characters:

- A: Capital letter
- a: Lower case letter
- 0: Number
- : Symbol character (asterisks, plus, minus, dots, slashes, any character that is not one of the others)

If, for example, in an HP center, the password needs to have at least 3 capital letters (3 or more), the template will have "AAA" in it. The order of these capital letters may be random. The same policy is followed for the rest of character types.

If for example the template is AAAaa00, the password MUST have at least 3 capital letters, at least two lower case letters, at least 2 numbers and at least one symbol. The password HpCodeWars21-

would be OK. It has 3 capital letters (H,C,W), at least two lower case letters, at least two numbers (21) and at least one symbol (the dash).

Note 1: Note that neither the password nor the template can have white spaces.

Note 2: The tests will use templates with only those 4 characters (A, a, 0 and).

Output

The program will return wether the password is robust or not. It will simply return the texts ROBUST or NOT ROBUST, depending on the template and the password passed in the input line.

Example 1

Input

```
AaaA0* HPcodewars2021-
```

Output

```
ROBUST
```

Example 2

Input

```
Aaa00AA00* HPcodewars2021-
```

Output

```
NOT ROBUST
```

The reason is because there are only two capital letters and at least 3 are required.

Solutions

Python

```
import re
def main():
    (pattern, password) = [i.strip() for i in input().split()]
    required_lower = pattern.count('a')
    required_upper = pattern.count('A')
    required_numbers = pattern.count('0')
    required_syms = pattern.count('*')
    actual_lower = len(re.findall("[a-z]", password))
    actual_upper = len(re.findall("[A-Z]", password))
    actual_numbers = len(re.findall("[0-9]", password))
    actual_syms = len(password) - actual_lower - actual_upper -
actual_numbers
    if actual_lower >= required_lower and actual_upper >= required_upper and
actual_numbers >= required_numbers and actual_syms >= required_syms:
        print("ROBUST")
    else:
        print("NOT ROBUST")
if __name__ == "__main__":
    main()
```

C++

```
#include <string>
#include <iostream>
#include <regex>
#include <ctype.h>
int main(int argc, char **argv)
{
    std::string inputted;
    getline(std::cin, inputted);
    std::stringstream ss(inputted);
    std::string s;
    std::vector<std::string> out;
    bool invalid=false;
    int numMays=0, numMins=0, numNums=0, numSyms=0;
    int realNumMays=0, realNumMins=0, realNumNums=0, realNumSyms=0;
    //Extract template and password
    while (std::getline(ss, s, ' ')) {
        out.push_back(s);
    }
    //Check only two section on input
    if (out.size()!=2)
    {
        std::cout << "invalid number of parameters:" << out.size() <<
std::endl;
```

```
        return -1;
    }
    std::string ptemplate=out[0];
    std::string pass=out[1];
    //check if template is OK
    for (int i = 0; i < ptemplate.length(); i++) {
        if (ptemplate.at(i) == 'A') numMays++;
      if (ptemplate.at(i) == 'a') numMins++;
      if (ptemplate.at(i) == '0') numNums++;
      if (ptemplate.at(i) == '*') numSyms++;
    }

    //Check robustness of password
    for (int i = 0; i < pass.length(); i++) {
        if (isupper(pass[i])) realNumMays++;
      else if (islower(pass[i])) realNumMins++;
      else if (isdigit(pass[i])) realNumNums++;
      else realNumSyms++;
    }
    if (realNumMays>=numMays && realNumMins>= numMins && realNumNums>=numNums
&& realNumSyms>=numSyms)
          std::cout << "ROBUST";
    else
          std::cout << "NOT ROBUST";
    return 0;
}
```

22 Dice Roller Plus

13 points

Introduction

In HP we love board games. But there's always somebody who losses their dices and cannot play anymore.

We need somebody who write a program that throws virtual dices in order to play board games. Every game is different and may require from 2 to 5 dices. And every dice is different, they may have from 4 to 20 faces (numbered consecutively from 1 to 4 or from 1 to 20...).

But... pssst, only between you and me... let's modify the dice behaviour so that we can know the next dice score... Odd rolls will always increase by 3 while even rolls will always decrease by 2 in respect to the previous roll. Of course, if in a 6-face dice last roll was a 5 and you need to increase it by 3, it cannot be an 8, you need to start back from the lowest score: 4-5-6-1-2... so 5+3 means 2, or 1 - 2 means 5. This is an operation called "modulo", and its symbol is %. So (5+3) % 6=2.

But... remember: nobody should know about this trick.

Exercise

You are asked to write a program to obtain next dice roll, given a number of constraints. You will be given the number of dices and the number of faces of the dices. And then you will be given the last dice roll. And you need to write the next dice roll.

We also want to know the highest dice, the lowest dice and the total sum of the dices.

Input

The first input line will indicate the number of dices of the game and the number of faces of the dices. All dices have the same number of faces, separated by a comma:

```
<# dices>,<# faces>
```

The second input line will show the last dice roll, for all dices, separated by commas. Remember that this roll needs comply with the number of dices and the number of faces:

```
<dice 1 roll>,<dice 2 roll>,...,<dice N roll>
```

Output

The output will show next dice roll given that odd rolls (first, third, fifth...) will increase by 3 modulo num_faces while even rolls (second, fourth...) will decrease by 2 modulo num_faces. The output line will separate dice results by an space, then it will have a dot, another space and then the highest dice, the lowest dice and the total sum of the dices.

```
13 2 15 11 2 . Max value is 15 in dice 3. Min value is 2 in dice 2. The total
sum is 48.
```

Note: If two dices have the same min value or the same max value, the indicated position will be the first one.

Example 1

Input

```
5,15                       <---- This means 5 dices of 15 faces each
13,2,15,11,7               <---- This is the last roll
```

Output

```
1 15 3 9 10 . Max value is 15 in dice 2. Min value is 1 in dice 1. The total
sum is 38.
```

This is because first roll (odd) was a 13, so 13+3=16, which is 1 modulo 15. Second roll (even) was a 2, so 2-2=0, which corresponds to 15 (because dice start at 1). Third roll (odd) was a 15, so 15+3=18, which is 3 modulo 15. Fourth roll (even) was a 11, so 11-2=9. Fifth roll (odd) was a 7, so 7+3=10.

Example 2

Input

```
3,6
5,4,1
```

Output

```
2 2 4 . Max value is 4 in dice 3. Min value is 2 in dice 1. The total sum is
8.
```

Solutions

Python

```python
#!/usr/bin/env python3
def is_odd(number):
    return bool(number % 2)
def is_even(number):
    return not(is_odd(number))
```

```
if __name__ == "__main__":
    dice_desc=input("")
    last_roll=input("")
    try:
        dices_n = int(dice_desc.split(',')[0])
        dices_faces = int(dice_desc.split(',')[1])
        rolls = list(map(int,last_roll.split(',')))
    except:
        print("The input is invalid.")
        exit(-1)
    if len(rolls) != dices_n:
        print(f"Hey, that's an invalid rol. You are playing with {dices_n}
dices but only entered {len(rolls)}")
        exit(-1)
    result = []
    for i,current_dice in enumerate(rolls):
        if is_odd(i):
            calc= current_dice-2 if current_dice-2 >0 else
dices_faces+current_dice-2
            result.append(calc)
        else:
            sol=(current_dice+3)%dices_faces
            if sol==0:
                sol=dices_faces
            result.append(sol)
    max_val=max(result);
    max_pos=result.index(max_val);
    min_val=min(result);
    min_pos=result.index(min_val);
    sum_val=sum(result)
    final_str = " ".join([str(i) for i in result])
    print(f"{final_str} . Max value is {max_val} in dice {max_pos+1}. Min
value is {min_val} in dice {min_pos+1}. The total sum is {sum_val}.")
```

C++

```
#include <string>
#include <iostream>
#include <vector>
#include <sstream>
int main(int argc, char **argv)
{
    std::string input_text1="";
    std::cin >> input_text1;
    std::vector<int> vect;
    std::vector<int> vect2;
    std::stringstream s1(input_text1);
    for (int i; s1 >> i;) {
        vect.push_back(i);
        if (s1.peek() == ',')
            s1.ignore();
    }
    int num_dices=vect[0];
    int num_faces=vect[1];
    //std::cout << num_dices << "--" << num_faces << std::endl;
    std::string input_text2="";
    std::cin >> input_text2;
    std::stringstream s2(input_text2);
    for (int j; s2 >> j;) {
        vect2.push_back(j);
        if (s2.peek() == ',')
            s2.ignore();
    }
    if (vect2.size()!=num_dices)
    {
        std::cout << "ERROR";
        return -1;
    }
    for (int k=0;k<vect2.size(); k++)
    {
        if (k%2==0)
```

```
      {
           vect2[k]=(vect2[k]+3)%num_faces;
      }
      else
      {
           int extra;
           if (vect2[k]-2 <= 0 )
         {
               extra=num_faces;
         }
         else
         {
               extra=0;
         }

         vect2[k]=extra+vect2[k]-2;
      }
      std::cout << vect2[k] << " ";
    }
    std::cout << ".";
    int smallest_element = vect2[0];
    int largest_element = vect2[0];
    int sum_of_elements = vect2[0];
    int largest_pos=1;
    int smallest_pos=1;
    for(int i = 1; i < vect2.size(); i++)  //start iterating from the second
element
    {
        if (vect2[i] < smallest_element)
        {
            smallest_element = vect2[i];
            smallest_pos=i+1;
        }

      if (vect2[i] > largest_element)
```

```
        {
            largest_element = vect2[i];
          largest_pos=i+1;
        }

      sum_of_elements += vect2[i];
    }
    std::cout << " Max value is " << largest_element << " in dice " <<
largest_pos <<". Min value is " << smallest_element << " in dice " <<
smallest_pos << ". The total sum is " << sum_of_elements << ".";
    return 0;
}
```

23 The Hidden Sentence

13 points

Introduction

In the planet Digitus, although they use the same dictionary as ours, they have invented a way to code small messages with up to 10 different letters using multiplications of numbers. In order to do so, first they provide a multiplication whose result has all the needed characters, then its meaning, and then a sentence encoded with each word being a different multiplication.

In example:

```
34x313
queso
6x107
8x8
17x626
```

As you can see, the first result is 34x313 which is 10642. And that translates to "queso". That means that the "1" encondes the "q", the "0" encodes the "u" and so on. And the full sentence will be "eso es queso", talking into account the three multiplications that follow.

Exercise

Create a program that, given an input like the one above, returns the decoded sentence. The program will keep reading multiplications (words) until no more are provided (EOF).

Input

Initial multiplication, meaning, and then list of multiplications to decode. Multiplication factors are separated by the "x" character.

Output

Decoded sentence.

Example 1

Input

```
34x313
queso
6x107
```

```
8x8
17x626
```

Output

```
eso es queso
```

Example 2

Input

```
6679x80
remesa
150x29
32x135
```

Output

```
mera mesa
```

Solutions

Python

```
import sys
def calculate_multiplication(multiplication):
    op1, op2 = multiplication.split('x')
    return int(op1) * int(op2)
def main():
    initial_multiplication = input().strip()
    help_sentence = input().strip()
    initial_result = calculate_multiplication(initial_multiplication)
    initial_result_str = str(initial_result)
    translation = {}
    for i in range(len(help_sentence)):
        translation[initial_result_str[i]] = help_sentence[i]
    multiplication_number = 0
    for multiplication in sys.stdin:
        if multiplication_number > 0:
            print(' ', end='')
        result = str(calculate_multiplication(multiplication))
        for i in range(len(result)):
            print(translation[result[i]], end='')
        multiplication_number += 1
    print()
if __name__ == "__main__":
    main()
```

C++

```
#include <iostream>
#include <string>
#include <unordered_map>
using namespace std;
int calculate_multiplication(const string &multiplication)
{
    int op1, op2;
    string::size_type pos = multiplication.find('x');
    op1 = stoi(multiplication.substr(0, pos));
    op2 = stoi(multiplication.substr(pos + 1));
    return op1 * op2;
}
int main()
{
    string initial_multiplication, help_sentence;
    cin >> initial_multiplication >> help_sentence;
    int initial_result = calculate_multiplication(initial_multiplication);
    string initial_result_str = to_string(initial_result);
    unordered_map<char, char> translation;
    for (int i = 0; i < initial_result_str.length(); i++)
    {
        translation[initial_result_str[i]] = help_sentence[i];
    }
    int multiplication_number = 0;
    string multiplication;
    while (cin >> multiplication)
    {
        if (multiplication_number > 0)
```

```
            cout << ' ';
        string result = to_string(calculate_multiplication(multiplication));
        for (int i = 0; i < result.length(); i++)
        {
            cout << translation[result[i]];
        }
        multiplication_number++;
    }
    cout << endl;
    return 0;
}
```

24 Chess checker

15 points

Introduction

Chess is an ancient game with strict rules. The chessboard is composed of 64 squares and each player has 8 pawns, two rooks, two knights, two bishops, a queen and the king.

Exercise

We ask you to write a program that, given a board with its chess pieces, indicates whether the board is a valid game or not.

It's important to remember that:

- There are as much as 8 pawns
- There MUST be a king of every color
- Pawns cannot be in the first nor last row of the board

NOTE: it is actually possible to have more than two rooks, two knights, two bishops, and a queen per colour, as they can be changed when the pawns reach the last row, BUT in this case the numbers of pawn should be reduced correspondingly. This has to be checked as well.

Input

The input will be eight lines.

Each line will contain a number of capital letters indicating a chess piece (or empty space), no longer than 8 letters.

The letters for the pieces and the empty spaces are:

```
0 --> Blank Square
P --> White Pawn
T --> White Rook
C --> White Knight
A --> White Bishop
R --> White Queen
Y --> White King
W --> Black Pawn
```

```
Z --> Black Rook
N --> Black Knight
B --> Black Bishop
Q --> Black Queen
K --> Black King
```

Output

The output will be a single line indicating either

```
VALID
```

or

```
NOT VALID
```

Example 1

Input

```
TCARYACT
PPPPPPPP
00000000
00000000
00000000
00000000
WWWWWWWW
ZNBQKBNZ
```

Output

```
VALID
```

Example 2

Input

```
TCARYACT
P0000PPP
00000000
00000000
00000000
00000000
PP00000P
```

```
ZNNKKBNZ
```

Output

```
NOT VALID
```

Because there are two black kings.

Solutions

Python

```python
import sys
rows = []
verify = ["0", "P", "T", "C", "A", "R", "Y", "W", "Z", "N", "B", "Q", "K"]
for i in range(8):
    row = input().strip()
    rows.append(row)
for i in range(len(rows)):
    if len(rows[i]) > 8 or len(rows[i]) < 8 or not rows[i]:
        sys.exit("ERROR")
    row = str(rows[i])
    v = False
    for x in verify:
        if row.find(x) != -1:
            v = True
            break
    if v == False:
        sys.exit("ERROR")
count = [0] * 12
for row in rows:
    for r in row:
        if r == "P":
            count[0] += 1
        if r == "T":
            count[1] += 1
        if r == "C":
            count[2] += 1
        if r == "A":
            count[3] += 1
        if r == "R":
            count[4] += 1
        if r == "Y":
            count[5] += 1
        if r == "W":
            count[6] += 1
        if r == "Z":
            count[7] += 1
        if r == "N":
            count[8] += 1
        if r == "B":
            count[9] += 1
        if r == "Q":
            count[10] += 1
        if r == "K":
            count[11] += 1
if count[0] > 8 or count[6] > 8 or count[5] != 1 or count[11] != 1:
    print("NOT VALID")
    sys.exit()
replaced = 0
if count[1] > 2:
    replaced = replaced + count[1]-2
if count[2] > 2:
    replaced = replaced + count[2]-2
if count[3] > 2:
    replaced = replaced + count[3]-2
if count[4] > 1:
    replaced = replaced + count[4]-1
if (count[0] + replaced) > 8:
    print("NOT VALID")
    sys.exit()
replaced = 0
if count[7] > 2:
    replaced = replaced + count[7]-2
```

```
if count[8] > 2:
    replaced = replaced + count[8]-2
if count[9] > 2:
    replaced = replaced + count[9]-2
if count[10] > 1:
    replaced = replaced + count[10]-1
if (count[6] + replaced) > 8:
    print("NOT VALID")
    sys.exit()
invalid = False
for i in range(len(rows)):
    for j in range(len(rows[i])):
        if rows[0][i] == "P":
            invalid = True
        if rows[0][i] == "W":
            invalid = True
        if rows[7][i] == "P":
            invalid = True
        if rows[7][i] == "W":
            invalid = True
if invalid == True:
    print("NOT VALID")
else:
    print("VALID")
```

C++

```
#include <string>
#include <iostream>
#define boardSize 8
int main(int argc, char **argv)
{
    std::string rows[boardSize];
    bool invalid=false;
    long count[12];
    for (int i=0; i<boardSize; i++)
    {
        std::cin >> rows[i];
    }
    for (int i=0; i<boardSize; i++)
    {
        if ((rows[i].length()>boardSize) || (rows[i].length()<boardSize) ||
rows[i].empty())
        {
            std::cout << "ERROR";
            return -1;
        }
        //#Look for strange characters
        if ((rows[i].find_first_not_of("0PTCARYWZNBQK") !=
std::string::npos))
        {
            std::cout << "ERROR";
            return -1;
        }
    }
    //Counters
    for (int i=0; i<12; i++) count[i]=0;
    for (int i = 0; i < boardSize; i++)
    {
       for (int j = 0; j < boardSize; j++)
       {
           if (rows[i][j] == 'P') count[0]++;
           if (rows[i][j] == 'T') count[1]++;
           if (rows[i][j] == 'C') count[2]++;
           if (rows[i][j] == 'A') count[3]++;
           if (rows[i][j] == 'R') count[4]++;
           if (rows[i][j] == 'Y') count[5]++;
           if (rows[i][j] == 'W') count[6]++;
           if (rows[i][j] == 'Z') count[7]++;
           if (rows[i][j] == 'N') count[8]++;
           if (rows[i][j] == 'B') count[9]++;
```

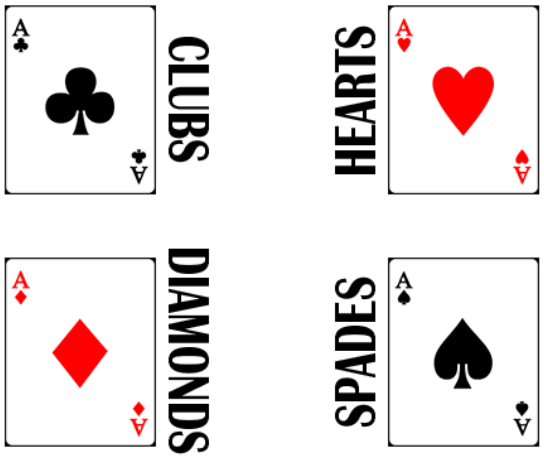
```
            if (rows[i][j] == 'Q') count[10]++;
            if (rows[i][j] == 'K') count[11]++;
        }
    }
    //Count pawns and kings
    if (count[0]>8 || count[6]>8 || count[5]!=1 || count[11]!=1)
    {
            std::cout<<"NOT VALID";
            return 0;
    }
    //Count changed pieces
    int replaced=0;
    if (count[1]>2) replaced=replaced+count[1]-2;
    if (count[2]>2) replaced=replaced+count[2]-2;
    if (count[3]>2) replaced=replaced+count[3]-2;
    if (count[4]>1) replaced=replaced+count[4]-1;
    if ((count[0]+replaced) > 8)
    {
            std::cout<<"NOT VALID";
            return 0;
    }
    replaced=0;
    if (count[7]>2) replaced=replaced+count[7]-2;
    if (count[8]>2) replaced=replaced+count[8]-2;
    if (count[9]>2) replaced=replaced+count[9]-2;
    if (count[10]>1) replaced=replaced+count[10]-1;
    if ((count[6]+replaced) > 8)
    {
            std::cout<<"NOT VALID";
            return 0;
    }
    //Pawn Positions in last rows
    invalid=false;
    for (int i = 0; i < boardSize; i++)
    {
        if (rows[0][i] == 'P') invalid=true;
        if (rows[0][i] == 'W') invalid=true;
        if (rows[boardSize-1][i] == 'P') invalid=true;
        if (rows[boardSize-1][i] == 'W') invalid=true;
    }
    if (invalid)
    {
        std::cout<<"NOT VALID";
        return 0;
    }
    std::cout<<"VALID";
return 0;
}
```

25 Cardpile Sorting

16 points

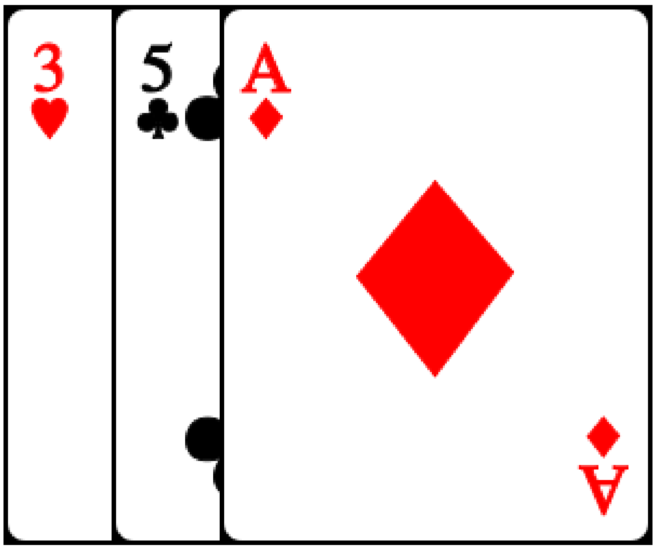
Introduction

You're given a randomly sorted american 44-card deck. Every card in the deck has a suit (Hearts, Diamonds, Spades, and Clubs), and a number (A, 2, 3, 4, 5, 6, 7, 8, J, Q, K). The objective is to sort the deck on 4 stacks, one for each suit on the deck (one pile for all the hearts, one pile for all the diamonds, and so forth)



However, there are some rules on how to play:

- The entire deck is upside down.
- To draw, you take the first three cards on the deck, and put them in draw order one on top of each other. For example, in the following picture, if the first three cards of the deck were the Three of Hearts, the Five of Clubs, and the Ace of Diamonds you'd put them like this:



- You can only move to a pile, a card that doesn't have another one on top (in the previous example, the Ace of Diamonds)
- You can only build the piles in order (A, 2, 3, 4...). The Ace has to be the first card.
- If a card that doesn't have another one on top can be moved to its respective pile, it will be moved to said pile automatically.
- If we cannot move a card to a pile, we will then draw the following three cards and repeat the process (in the previous example, we put the top card on top of the ace of diamonds and so forth)
- If we have to draw cards and there are less than 3 cards left on the deck, we draw as many cards as there are available.
- If we have to draw cards and there are NO cards left on the deck, we take the face up cards, and group them again in the same order without shuffling, and use it as the deck to repeat the process.
- If we go through an entire lap of drawing the deck and we cannot move a single card, we lose.

- If we manage to build the four piles, we win.

Exercise

You are asked to write a write a program that decides whether or not any given deck will beat the card pile challenge.

The program will take the input in a single line, that contains the pre-shuffled deck.

The input will contain every single card separated by dashes.

Every card will have two characters, the first one for the number (A, 2, 3, 4, 5, 6, 7, 8, J, Q, K), and the second one for the suit (S for Spade, H for Heart, D for Diamond and C for Club respectively).

If the input is valid, the output should be:

```
We beat the game
```

Otherwise:

```
We didn't beat the game
```

Input

List of cards in the deck, with number and suit.

Output

Whether the user wins the game or not.

Example 1

Input

```
AS-AH-AD-2S-2H-2D-3S-3H-3D-4S-4H-4D-5S-5H-5D-6S-6H-6D-7S-7H-7D-8S-8H-8D-JS-
JH-JD-QS-QH-QD-KS-KH-KD-3C-2C-AC-6C-5C-4C-JC-8C-7C-KC-QC
```

Output

```
We beat the game
```

Example 2

Input

```
AS-AH-AD-2S-2H-2D-3S-3H-3D-4S-4H-4D-5S-5H-5D-6S-6H-6D-7S-7H-7D-8S-8H-8D-JS-
JH-JD-QS-QH-QD-KS-KH-KD-3C-AC-2C-6C-5C-4C-JC-8C-7C-KC-QC
```

Output

```
We didn't beat the game
```

Solutions

Python

```
VALUE_MAPPER = {
    'A': 1,
    '2': 2,
    '3': 3,
    '4': 4,
    '5': 5,
    '6': 6,
    '7': 7,
    '8': 8,
    'J': 9,
    'Q': 10,
    'K': 11
}
def parse_input():
    raw_cards = input().split('-')
    deck = [(card[1], VALUE_MAPPER[card[0]]) for card in raw_cards]
    return deck
def all_piles_complete(piles):
    for pile in piles:
        if piles[pile] != 11:
            return False
    return True
def main():
    deck = parse_input()
    piles = {
        'C': 0,
        'H': 0,
        'D': 0,
        'S': 0
    }
    playing = True
    victory = False
    card_played_in_round = False
    new_deck = []
    while (playing):
        draw = deck[:3]
        draw.reverse()
        deck = deck[3:]
        cards_placed = 0
        for card in draw:
            suit = card[0]
            value = card[1]
            if piles[suit] == value - 1:
                piles[suit] = value
                card_played_in_round = True
                cards_placed += 1
            else:
                break
        draw.reverse()
        for i in range(0, len(draw)-cards_placed):
            new_deck.append(draw[i])
        if all_piles_complete(piles):
            victory = True
            playing = False
        if len(deck) == 0 and not card_played_in_round:
            playing = False
        if len(deck) == 0 and card_played_in_round:
            deck = new_deck
            new_deck = []
            card_played_in_round = False
    if victory:
        print("We beat the game")
    else:
        print("We didn't beat the game")
if __name__ == "__main__":
    main()
```

C++

```cpp
#include <iostream>
#include <vector>
#include <sstream>
#include <map>
using namespace std;
int main(int argc, char *argv[])
{
    string inputString, card;

    //Make an array of strings with the input
    cin >> inputString;
    stringstream stream(inputString);

    vector<string> input;
    while(getline(stream, card, '-')){
        input.push_back(card);
    }

    //Map to convert characters to integers
    map<char, int> charConversor = {
        {'A', 1}, {'2', 2}, {'3', 3}, {'4', 4}, {'5', 5}, {'6', 6},
        {'7', 7}, {'8', 8}, {'J', 9}, {'Q', 10}, {'K', 11}
    };

    map<char, int> piles = {
        {'S', 0}, {'D', 0}, {'H', 0}, {'C', 0}
    };

    //Define the necessary variables
    int currentCardIndex = -1;              //To iterate through our deck
    bool stillPlaying = true;               //Main game loop
    bool isVictory = true;                  //To check our win condition
    bool hitACardThisLap = false;           //Safety valve, to know if we
moved a card to a pile on an entire deck lap

    //Play the game
    while(stillPlaying)
    {

        //Move the index three cards forward
        int deckSize = input.size();
        currentCardIndex = (currentCardIndex + 3 > deckSize - 1) ? deckSize -
1 : currentCardIndex + 3;

        //Iterate the face up pile, a.k.a check the top card of the upside-
down subdeck
        //from the current card, to the first
        bool moveCards = true;
        if(currentCardIndex >= 0)
        {
            //Repeat while the 'top card' is movable and we don't have to
draw
            while(moveCards)
            {
                char numberChar = input[currentCardIndex][0];
                char suit = input[currentCardIndex][1];
                int numberInt = charConversor[numberChar];

                //If the pile pile of this card accepts it, move it
                if(piles[suit] + 1 == numberInt)
                {
                    piles[suit] = piles[suit] + 1;
                    input.erase(input.begin() + currentCardIndex); // 'Moved
to the pile'
                    hitACardThisLap = true;
                    --currentCardIndex;
                }
                else
                {
                    moveCards = false;
                }

                if(currentCardIndex < 0)
                {
                    moveCards = false;
                }
            }
        }

        //Check if there are remaining cards in deck, otherwise flip the deck
        if(currentCardIndex == (input.size() - 1))
        {
            currentCardIndex = -1;

            //If we can't do any more moves, end the game
            if(!hitACardThisLap)
            {
                stillPlaying = false;
                isVictory = input.size() == 0;
            }

            hitACardThisLap = false;
        }
    }

    //Check the result
    if(isVictory)
    {
        cout << "We beat the game" << endl;
    }
    else
    {
        cout << "We didn't beat the game" << endl;
    }
}
```

26 The flight attendant trick

18 points

Introduction

A new COVID regulation asks flight attendants to memorize the positions of travellers in a flight to avoid seat permutations in small flights. A flight attendant has improved a trick consisting in memorize two numbers as short as possible, depending on the location of travellers.

To implement this trick, the flight attendant counts the positions of men and women along the plane, and ends up with a number of no more than 17 digits or letters. To obtain it, they follow several rules. First, the attendant looks over the plane counting how many men and women are sitting together at the beginning of the flight.

For example, with a plane like the following (being H a man and M a woman):

```
HM MM
HH HM
MM HH
MH MH
HH HH
MM MM
MH HH
HM MH
HM MM
HH HM
```

The calculated number by the attendant is 1H 3M 3H 3M 2H 1M 1H 1M 5H, etc, that would result in a number like 133321155422331.

To try to make the number to be as short as possible to memorize it, the attendant makes a second pass counting the travellers in longitudinal order as well. In this case the obtained number is 2H 2M 1H 2M 3H 1M 1H 1M 2H 1M 1H... which means 2212311211211211211112123122.

As you can see for this trick "together" doesn't mean exactly in the same row or column, it also "overflows" to the next one (like in the first example above, the second group of 3 women).

The second step of the trick is to give it another folding and make the following conversion:

```
111 --> A
```

```
112 --> B
121 --> C
122 --> D
211 --> E
212 --> F
221 --> G
222 --> H
```

This conversion is done starting from the beginning of the number obtained in first step.

This makes the number to be:

```
133321155422331 --> 1333E55422331
or
2212311211211211211112123122 --> G23BBBBAC23
```

Exercise

As not all flight attendants are able to memorize this process, the company is asking you to develop a program that shows the shortest number result from these two options.

Finally, the gender of the first passenger will be indicated at the beginning of the output with an *X* for a female and a *Y* for a male.

Input

The input is the location of travellers along a flight of no more than 10 rows of seats with 4 seats per row. An input with more than 10 rows or rows different to 4 seats would return an error (as explained in the output section) and will not be considered in this exercise.

Rows will only have capital letters H and M, for men and women respectively, with no spaces in between them.

To number of rows may be variable from 1 to 10 and to indicate the end of the data, a line with a dot will be provided.

Output

A line representing the shortest obtained number given the previously explained process. If both options (transversal and longitudinal counting) return an equal-length string, the tool will return the transversal one.

An input with more than 10 rows or rows different to 4 seats would return the text:

```
ERROR
```

and will finish the program.

Example 1

Input

```
HMMM
HHHM
MMHH
MHMH
HHHH
MMMM
MHHH
HMMH
HMMM
HHHM
.
```

Output

```
YG23AEEEB123D
```

This is because:

Transversal code is: 1333E155422331 (14 chars).

Longitudinal code is: G23AEEEB123D (12 chars).

Example 2

Input

```
HMMM
HHHH
HMHH
MHHH
MMMM
MHHH
HMMH
HMMM
```

```
HHHM
.
```

Output

```
Y135C35422331
```

This is because:

13512135422331 --> 135C35422331 (12 chars).

3331111112113112113122 --> 333AAE3B113D (12 chars).

Solutions

Python

```
def obtainCode(output, first):
    index = 0
    substring = ""
    exit = ""
    while index+3 <= len(output):
        substring = output[index:index+3]
        if substring == "111":
            output = output[:index] + "A" + output[index+3:]
        if substring == "112":
            output = output[:index] + "B" + output[index+3:]
        if substring == "121":
            output = output[:index] + "C" + output[index+3:]
        if substring == "122":
            output = output[:index] + "D" + output[index+3:]
        if substring == "211":
            output = output[:index] + "E" + output[index+3:]
        if substring == "212":
            output = output[:index] + "F" + output[index+3:]
        if substring == "221":
            output = output[:index] + "G" + output[index+3:]
        if substring == "222":
            output = output[:index] + "H" + output[index+3:]
        index += 1
    if first == "H":
        exit = "Y"
    else:
        exit = "X"
    exit += output
    return exit
def main():
    rows = [None] * 12
    output = ""
    output2 = ""
    last = ""
    num = 0
    count = 0
    inputted = ""
    i = 0
    while inputted != ".":
        inputted = input()
        rows[i] = inputted
        i += 1
    numrows = i - 1
    for i in range(numrows):
        if len(rows[i]) > 4 or len(rows[i]) < 4 or not rows[i]:
            print("ERROR")
            return
        if any([c not in "HM" for c in rows[i]]):
            print("ERROR")
            return
    last = rows[0][0]
    for i in range(numrows):
        for j in range(4):
            if rows[i][j] == last:
                count += 1
            else:
                output += str(count)
                count = 1
                last = rows[i][j]
    output += str(count)
    last = rows[0][0]
    count = 0
    for j in range(4):
        for i in range(numrows):
            if rows[i][j] == last:
                count += 1
            else:
                output2 += str(count)
                count = 1
                last = rows[i][j]
    output2 += str(count)
    output = obtainCode(output, rows[0][0])
    output2 = obtainCode(output2, rows[0][0])
    if len(output) <= len(output2):
        print(output)
    else:
        print(output2)
if __name__ == "__main__":
    main()
```

C++

```
#include <string>
#include <iostream>
#include <regex>
std::string obtainCode(std::string output, char first)
{
    size_t index = 0;
    std::string substring;
    std::string exit;
    while (index+3<=output.length()) {
        substring=output.substr(index, 3);
        if (substring=="111") output.replace(index, 3, "A");
        if (substring=="112") output.replace(index, 3, "B");
        if (substring=="121") output.replace(index, 3, "C");
        if (substring=="122") output.replace(index, 3, "D");
        if (substring=="211") output.replace(index, 3, "E");
        if (substring=="212") output.replace(index, 3, "F");
        if (substring=="221") output.replace(index, 3, "G");
        if (substring=="222") output.replace(index, 3, "H");
        index++;
    }
    //#MALE - FEMALE (X/Y)
    if (first=='H')
    {
       exit="Y";
    }
    else
    {
       exit="X";
    }
    exit.append(output);
    return exit;
}
int main(int argc, char **argv)
{
    std::string rows[12];
    std::string output, output2;
    char last;
    long num=0;
    long count=0;
    //INPUT
    std::string inputted="";
    int i=0;
    while (inputted!=".")
    {
            std::cin >> inputted;
            rows[i]=inputted;
            i++;
```

```
}
int numrows=i-1;
//#CHECKS
for (int i=0; i<numrows; i++)
{
    if ((rows[i].length()>4) || (rows[i].length()<4) || rows[i].empty())
    {
        std::cout << "ERROR";
        return 0;
    }
    if ((rows[i].find_first_not_of("HM") != std::string::npos))
    {
        std::cout << "ERROR";
        return 0;
    }
}
//#COUNT1
last=rows[0][0];
for (int i=0; i<numrows; i++)
{
    for (int j=0; j<4; j++)
    {
        if (rows[i][j]==last)
        {
            count++;
        }
        else
        {
            output.append(std::to_string(count));
            count=1;
            last=rows[i][j];
        }
    }
}
output.append(std::to_string(count));
```

```
    //#COUNT2
    last=rows[0][0];
    count=0;
    for (int j=0; j<4; j++)
    {
        for (int i=0; i<numrows; i++)
        {
            if (rows[i][j]==last)
            {
                count++;
            }
            else
            {
                output2.append(std::to_string(count));
                count=1;
                last=rows[i][j];
            }
        }
    }
    output2.append(std::to_string(count));
    output=obtainCode(output,rows[0][0]);
    output2=obtainCode(output2,rows[0][0]);
    if (output.length()<=output2.length())
    {
        std::cout<<output;
    }
    else
    {
        std::cout<<output2;
    }
    return 0;
}
```

27 Intergalactic Excel

19 points

Introduction

Elon Musk has decided to create an interplanetary accounting system and he wants to use Excel for it. However, as the bitrate in interplanetary communications is quite scarce, he has decided to dispense with a graphical interface and he wants to implement it in a simplified way and in text mode.

Furthermore, as the guy is short on money, he has decided to ask for help from some kids from the high school of his friend's nieces and it turns out that you are those kids.

Exercise

You are asked to write a program that reads 5 lines of text simulating an Excel spreadsheet. As you know, each cell of an Excel spreadsheet is numbered according to its row and column. So for example, cell C2 corresponds to column 3 and row 2. In the Excel that is to be implemented, it is the same. The cells are separated by commas and for this proof of concept, it will only have 5 rows and 5 columns.

Cells can be empty, have numbers or have operations. For this proof of concept, the operations will be sums, subtractions or multiplications.

Output should be the same Excel sheet received, but with the operations solved. For example:

```
0,4,5,,
3+1,,5*4,,
,,,,
7*3,,,,
,,1-3,,
```

The output should be:

```
0,4,5,,
4,,20,,
,,,,
21,,,,
,,-2,,
```

However, things are not going to be that easy. Mr. Musk is also asking us for the Excel to be able to do "formulas". That is, in the operations we must be able to reference other cells. So if a cell appears as *$XY*, where X is a letter and Y is a number, we must read the referenced cell and substitute that *$XY* for the number referenced. For example:

```
0,4,5,,
3+6,,$A1*$A2,,
$A2*2,,,,5+$A2
,,,,
,$B1,,,
```

Should have this output:

```
0,4,5,,
9,,0,,
8,,,,9
,,,,
,9,,,
```

As said, for cells the operations will also be sum, substraction and multiplication. Also, the maximum number of cells that can be referenced is 2, but they can also appear involved in an operation with an escalar number (see above) or by themselves, just referencing the value of the cell.

Another limitation is that **there will be no chain references**. If a cell references another cell, that cell will either have a value or an operation between scalars, but not a reference to another cell.

Input

Five lines of text with the Excel spreadsheet, containing empty cells, values and operations, organized in 5 columns separated by commas.

Output

Five lines of text with the Excel spreadsheet with the operations solved and formulas resolved, organized in 5 columns separated by commas.

Example 1

Input

```
0,4,5,,
3+1,,5*4,,
,,,,
7*3,,,,
,,1-3,,
```

Output

```
0,4,5,,
4,,20,,
,,,,
21,,,,
,,-2,,
```

Example 2

Input

```
0,4,5,,
3+6,,$A1*$A2,,
$A2*2,,,,5+$A2
,,,,
,$B1,,,
```

Output

```
0,4,5,,
9,,0,,
8,,,,9
,,,,
,9,,,
```

Solutions

Python

```
import re
def get_numbers_operation(value):
    try:
        match = re.search(r'^(\d+)([\+\-\*])(\d+)$', value)
    except:
        return None, None, None
    if match:
        return match.groups()
    else:
        return None, None, None
def calculate_operation(n1, op, n2):
    if op == '+':
        return n1 + n2
    elif op == '-':
        return n1 - n2
    elif op == '*':
        return n1 * n2
def evaluate_operations(spreadsheet):
    # this mutates spreadsheet
    for row in range(5):
        for column in range(5):
            value = spreadsheet[row][column]
            n1, op, n2 = get_numbers_operation(value)
            if n1 and op and n2:
                spreadsheet[row][column] = calculate_operation(
                    int(n1), op, int(n2))
    return spreadsheet
def replace_references_single(spreadsheet):
    # this mutates spreadsheet
    for row in range(5):
        for column in range(5):
            value = spreadsheet[row][column]
            if isinstance(value, str):
                match = re.search(r'(.*)(\$[A-E])(\d+)(.*)', value)
                if match:
                    column_ref = match.group(2)[1:]  # remove the $
                    column_index = ord(column_ref) - ord('A')
                    row_index = int(match.group(3)) - 1
                    spreadsheet[row][column] =
f"{match.group(1)}{spreadsheet[column_index][row_index]}{match.group(4)}"
    return spreadsheet
def print_spreadsheet(spreadsheet):
    for row in range(5):
        for column in range(5):
            print(spreadsheet[row][column], end=(',' if column < 4 else ''))
        print()
def main():
    spreadsheet = []
    # read the spreadsheet and store the raw values
    for _ in range(5):
        line = input().strip()
        values_in_line = line.split(',')
        spreadsheet.append(values_in_line)
    # first pass: evaluate simple operations
    spreadsheet = evaluate_operations(spreadsheet)
    # second pass: replace references
    spreadsheet = replace_references_single(spreadsheet)
    # third pass: replace references again (because one formula can involve
two references)
    spreadsheet = replace_references_single(spreadsheet)
    # fourth pass: evaluate operations resulting from formulas
    spreadsheet = evaluate_operations(spreadsheet)
    # now just print it
    print_spreadsheet(spreadsheet)
if __name__ == '__main__':
    main()
```

28 Diophantine Equations

20 points

Introduction

Diophantine equations are simply algebraic equations whose coefficients and solutions, if they exist, are integer numbers.

In particular, we are interested in the classical example: the lineal diophantine equation with two unknows, i.e., *ax + by = c* where *a*, *b* and *c*, and the unknown solutions (if they exist) *x* and *y* are integer numbers. In this case, the equation has solution if the greatest common divisor of *a* and *b* divides *c*, and when so the solutions are infinite.

For instance:

3x + 2y = -1

has a particular solution *x_p = -1* and *y_p = 1*. Then, the general solution is:

x = x_p + lambda *b / d = -1 + lambda* 2 / 1

y = y_p - lambda *a / d = 1 - lambda* 3 / 1

where *d* is the greatest common divisor of *a* and *b*, and *lambda* is any integer number. Therefore *x = -3* and *y = 4* (when *lambda = -1*); *x = 5* and *y = -8* (when *lambda = 3*), etc. are also solutions.

Exercise

Write a program that, for a given lineal diophantine equation with two unknows *ax + by = c* where *a*, *b* and *c* are different from zero, finds the particular solution closest to zero (i.e., when the sum of the absolute solution values is smaller: *abs(x) + abs(y)*) if it exists. In case there is more than one print them sorted. Otherwise, when the equation has no solution then print the following message:

There are no solutions.

Input

Three int numbers different from zero separated by *;* which correspond to *a*, *b* and *c* respectively:

```
a;b;c
```

Output

The particular solution closest to zero:

```
x;y
```

or the sorted particular solutions when there are more than one:

```
x0;y0
x1;y1
[...]
```

In case the equation has no solution then:

```
There are no solutions.
```

Example 1

Input

```
1;1;-1
```

Output

```
-1;0
0;-1
```

Example 2

Input

```
162;-508;5123866
```

Output

```
45;-10072
```

Solutions

Python

```python
def main() -> None:
    a, b, c = read_input()
    d = get_gcd(a, b)
    if c % d != 0:
        print("There are no solutions.")
        return
    a, b, c = a // d, b // d, c // d
    x0, y0, x1, y1 = solve_positive_equation(a, b, c)
    x0, y0 = add_signs_to_solution(x0, y0, a, b, c)
    solutions = [f"{x0};{y0}"]
    if a == b: solutions.append(f"{y0};{x0}")
    elif a == -1 * b: solutions.append(f"{-1 * y0};{x0}")
    if x1 != None and y1 != None and x1 != 0 and y1 != 0:
        x1, y1 = add_signs_to_solution(x1, y1, a, b, c)
        solutions.append(f"{x1};{y1}")
    for s in sorted(solutions): print(s)
def read_input() -> list:
    input_elements = input()
    assert ';' in input_elements
    input_numbers = [int(i) for i in input_elements.split(';')]
    assert len(input_numbers) == 3
    for i in input_numbers: assert i != 0
    return input_numbers
def get_gcd(a: int, b: int) -> int:
    if (b == 0): return abs(a)
    return get_gcd(b, a % b)
def solve_positive_equation(a: int, b: int, c: int) -> tuple:
    a, b, c = abs(a), abs(b), abs(c)
    x0, y0 = get_euler_particular_solution(a, b, c)
    x0, y0, x1, y1 = get_smaller_solution(x0, y0, a, b)
    return x0, y0, x1, y1
def get_euler_particular_solution(a: int, b: int, c: int) -> tuple:
    for i in range(abs(b)):
        if (c - a * i) % b == 0:
            y = (c - a * i) // b
            x = (c - b * y) // a
    return x, y

def get_smaller_solution(x0: int, y0: int, a: int, b: int) -> tuple:
    _lambda = -1 if x0 >= y0 else 1
    sum_abs = lambda x, y: abs(x) + abs(y)
    while True:
        x1, y1 = get_another_particular_solution(x0, y0, a, b, _lambda)
        if sum_abs(x0, y0) > sum_abs(x1, y1):
            x0, y0 = x1, y1
            x1, y1= None, None
        elif sum_abs(x0, y0) == sum_abs(x1, y1): break
        else:
            x1, y1 = None, None
            break
    return x0, y0, x1, y1
def get_another_particular_solution(x: int, y: int, a: int, b: int, _lambda:
int) -> tuple:
    x = x + _lambda * b
    y = y - _lambda * a
    return x, y

def add_signs_to_solution(x: int, y: int, a: int, b: int, c: int) -> tuple:
    sing_a, sing_b, sing_c = a // abs(a), b // abs(b), c // abs(c)
    x *= sing_a * sing_c
    y *= sing_b * sing_c
    return x, y
if __name__ == '__main__':
    main()
```

C++

```
#include <iostream>
#include <cmath>
#include <assert.h>
#include <tuple>
#include <string>
#include <algorithm>
int get_gcd(int a, int b){
    if (b == 0){
        return abs(a);
    }
    return get_gcd(b, a % b);
}
std::tuple<int, int> get_euler_particular_solution(int a, int b, int c){
    int x, y;
    for (int i = 0; i < abs(b); i++){
        if ((c - a * i) % b == 0){
            y = (int) (c - a * i) / b;
            x = (int) (c - b * y) / a;
        }
    }
    return std::make_tuple(x, y);
}
std::tuple<int, int> get_another_particular_solution(int x, int y, int a, int
b, int _lambda){
    x = x + _lambda * b;
    y = y - _lambda * a;
    return std::make_tuple(x, y);
}
std::tuple<int, int, int, int> get_smaller_solution(int x0, int y0, int a,
int b){
    int _lambda;
    x0 >= y0 ? _lambda = -1 : _lambda = 1;
    int x1 = 0, y1 = 0;
    while (true){
        std::tie(x1, y1) = get_another_particular_solution(x0, y0, a, b,
_lambda);
        if (abs(x0) + abs(y0) > abs(x1) + abs(y1)){
            x0 = x1, y0 = y1;
            x1 = 0, y1 = 0;
        } else if (abs(x0) + abs(y0) == abs(x1) + abs(y1)){
            break;
        } else {
            x1 = 0, y1 = 0;
            break;
        }
    }
    return std::make_tuple(x0, y0, x1, y1);
}
std::tuple<int, int, int, int> solve_positive_equation(int a, int b, int c){
    a = abs(a), b = abs(b), c = abs(c);
    int x0, y0, x1, y1;
    std::tie(x0, y0) = get_euler_particular_solution(a, b, c);
    std::tie(x0, y0, x1, y1) = get_smaller_solution(x0, y0, a, b);
    return std::make_tuple(x0, y0, x1, y1);
}
std::tuple<int, int> add_signs_to_solution(int x, int y, int a, int b, int
c){
    int sing_a = a / abs(a), sing_b = b / abs(b), sing_c = c / abs(c);
    x *= sing_a * sing_c;
    y *= sing_b * sing_c;
    return std::make_tuple(x, y);
}
int main(int argc, char **argv){
    int a = 0, b = 0, c = 0;
    auto scanned = std::scanf("%i;%i;%i", &a, &b, &c);
    assert (a != 0 && b != 0 && c != 0);

    int d = get_gcd(a, b);
    if (c % d != 0){
        std::cout << "There are no solutions." << std::endl;
        return 0;
    }
    a = (int) a / d, b = (int) b / d, c = (int) c / d;
    int x0, y0, x1, y1;
    std::tie(x0, y0, x1, y1) = solve_positive_equation(a, b, c);
    std::tie(x0, y0) = add_signs_to_solution(x0, y0, a, b, c);
    std::string solutions[3] = {std::to_string(x0) + ";" +
std::to_string(y0), "", ""};
    if (a == b){
        solutions[1] = std::to_string(y0) + ";" + std::to_string(x0);
    } else if (a == -1 * b){
        solutions[1] = std::to_string(-1 * y0) + ";" + std::to_string(x0);
    }
    if (x1 != 0 && y1 != 0){
        std::tie(x1, y1) = add_signs_to_solution(x1, y1, a, b, c);
        solutions[2] = std::to_string(x1) + ";" + std::to_string(y1);
    }
    sort(std::begin(solutions), std::end(solutions));
    for(auto i: solutions){
        if (i != ""){
            std::cout << i << std::endl;
        }
    }
    return 0;
}
```

29 The cryptogram issue

20 points

Introduction

A cryptogram is a mathematical puzzle in which various symbols or letters are used to represent numeric digits, typically in the form of a sum or equation. This type of verbal arithmetic is called a cryptogram or cryptarithm.

The classic example, published in the July 1924 issue of Strand Magazine by Henry Dudeney, is:

```
SEND

+

MORE

========

MONEY
```

Each letter can only be replaced by a different digit (from 0 to 9). In this example the value of the letters in the solution is:

```
D = 7

E = 5

M = 1

N = 6

O = 0

R = 8

S = 9

Y = 2
```

(as SEND + MORE = 9567 + 1085 = 10652, which is MONEY).

Note: For this example, there's more than one solution because with D = 1, E = 5, M = 0, N = 3, O = 8, R = 2, S = 7, Y = 6, the cryptogram is also valid (as SEND + MORE = 7531 + 0825 = 08356, which is MONEY).

In the tests used to check your code there will be only a possible solution.

Exercise

We ask you to write a program that receives three inputs (first addend, second addend and result):

```
SEND
MORE
MONEY
```

and returns a line like with the numeric values of the sum and the result all in a row:

```
9567+1085=10652
```

Constraints:

- Addends must be only letters and must be capital letters
- Addends cannot be longer than 9 letters
- Letter 'Ñ' is not allowed
- If there is no solution, the program will return the text "NO SOLUTION"

Input

The input will be three lines:

- The first line contains a capital letter word, no longer than 10 letters, with no 'Ñ', corresponding with the first addend
- The second line contains a capital letter word, no longer than 10 letters, with no 'Ñ', corresponding with the second addend
- The third line contains a capital letter word, no longer than 10 letters, with no 'Ñ', corresponding with the result of the sum

Output

A line representing the sum, using only numbers and the symbols '+' and '='.

NOTE: In some cases there are several solutions for a single cryptarithmetic. This will not be the case in the tests used to check your code.

Example 1

```
DAME
MAS
AMOR
```

Output

```
8931+394=9325
```

Example 2

Input

```
PAPA
MAMA
BEBE
```

Output

```
4141+3131=7272
```

Example 3

Input

```
TRES
DOS
CINCO
```

Output

```
5724+384=06108
```

Example 4

Input

```
LUIS
BEA
ALBA
```

Output

```
1790+342=2132
```

Solutions

Python

```python
import numpy as np
import sys
class Node(object):
    def __init__(self, c, v):
        self.c = c
        self.v = v
use = np.full(10, 0, dtype=int)
```

```
def check(nodeArr, count, s1, s2, s3):
    val1 = 0
    val2 = 0
    val3 = 0
    m = 1
    for s in s1[::-1]:
        for j in range(count):
            if nodeArr[j].c == s:
                break
        val1 += m * nodeArr[j].v
        m *= 10
    m = 1
    for s in s2[::-1]:
        for j in range(count):
            if nodeArr[j].c == s:
                break
        val2 += m * nodeArr[j].v
        m *= 10
    m = 1
    for s in s3[::-1]:
        for j in range(count):
            if nodeArr[j].c == s:
                break
        val3 += m * nodeArr[j].v
        m *= 10
    if val3 == (val1+val2):
        return True
    return False
def permutation(count, nodeArr, n, s1, s2, s3):
    if n == count-1:
        for i in range(10):
            if use[i] == 0:
                nodeArr[n].v = i
                if check(nodeArr, count, s1, s2, s3):
                    return True
        return False
    for i in range(10):
        if use[i] == 0:
            nodeArr[n].v = i
            use[i] = 1
            if permutation(count, nodeArr, n + 1, s1, s2, s3):
                return True
            use[i] = 0
    return False
def solveCryptographic(s1, s2, s3):
    count = 0
    freq = np.full(26, 0, dtype=int)
    for s in s1:
        freq[ord(s)-ord('A')] += 1
    for s in s2:
        freq[ord(s)-ord('A')] += 1
    for s in s3:
        freq[ord(s)-ord('A')] += 1
    for f in freq:
        if f > 0:
            count += 1
    if(count > 10):
        print("ERROR")
    nodeArr = np.empty(count, dtype=Node)
    for i in range(count):
        nodeArr[i] = Node('', 0)
    j = 0
    for i in range(26):
        if freq[i] > 0:
            nodeArr[j].c = chr(i + ord('A'))
            j += 1
    output = ''
    sol = permutation(count, nodeArr, 0, s1, s2, s3)
    if sol:
        for s in s1:
```

```
            for j in range(count):
                if nodeArr[j].c == s:
                    output += str(nodeArr[j].v)
        output += '+'
        for s in s2:
            for j in range(count):
                if nodeArr[j].c == s:
                    output += str(nodeArr[j].v)
        output += '='
        for s in s3:
            for j in range(count):
                if nodeArr[j].c == s:
                    output += str(nodeArr[j].v)
        print(output)
    return sol
def main():
    s1 = input()
    s2 = input()
    s3 = input()
    if solveCryptographic(s1, s2, s3) == False:
        sys.exit("NO SOLUTION")
if __name__ == "__main__":
    main()
```

C++

```
#include <iostream>
#include <string>
#include <vector>
using namespace std;
vector<int> use(10);
struct node{
        char c;
        int v;
};
int check(node * nodeArr, const int count, string s1, string s2, string s3)
```

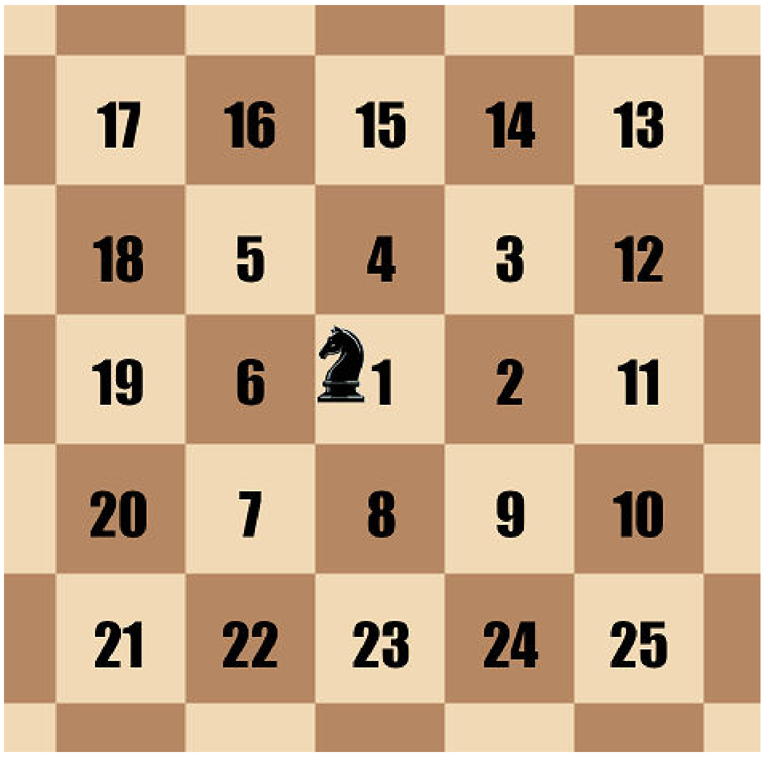
```
{
    int val1 = 0, val2 = 0, val3 = 0, m = 1, j, i;
    // calculate number corresponding to first string
    for (i = s1.length() - 1; i >= 0; i--)
    {
        char ch = s1[i];
        for (j = 0; j < count; j++)
            if (nodeArr[j].c == ch)
                break;
        val1 += m * nodeArr[j].v;
        m *= 10;
    }
    m = 1;
    // calculate number corresponding to second string
    for (i = s2.length() - 1; i >= 0; i--)
    {
        char ch = s2[i];
        for (j = 0; j < count; j++)
            if (nodeArr[j].c == ch)
                break;
        val2 += m * nodeArr[j].v;
        m *= 10;
    }
    m = 1;
    // calculate number corresponding to third string
    for (i = s3.length() - 1; i >= 0; i--)
    {
        char ch = s3[i];
        for (j = 0; j < count; j++)
            if (nodeArr[j].c == ch)
                break;
        val3 += m * nodeArr[j].v;
        m *= 10;
    }
    // sum of first two number equal to third return true
    if (val3 == (val1 + val2))
        return 1;
    // else return false
    return 0;
}
// Recursive function to check solution for all permutations
bool permutation(const int count, node* nodeArr, int n, string s1, string s2,
string s3)
{
    // Base case
    if (n == count - 1)
    {
        // check for all numbers not used yet
        for (int i = 0; i < 10; i++)
        {
            // if not used
            if (use[i] == 0)
            {
                // assign char at index n integer i
                nodeArr[n].v = i;
                // if solution found
                if (check(nodeArr, count, s1, s2, s3) == 1)
                {
//                      for (int j = 0; j < count; j++)
//                          cout << "" << nodeArr[j].c << "=" <<
nodeArr[j].v<<";";
                    return true;
                }
            }
        }
        return false;
    }
    for (int i = 0; i < 10; i++)
    {
        // if ith integer not used yet
        if (use[i] == 0)
        {
            // assign char at index n integer i
            nodeArr[n].v = i;
            // mark it as not available for other char
            use[i] = 1;
            // call recursive function
            if (permutation(count, nodeArr, n + 1, s1, s2, s3))
                return true;
            // backtrack for all other possible solutions
            use[i] = 0;
        }
    }
    return false;
}
bool solveCryptographic(string s1, string s2, string s3)
{
    // count to store number of unique char
    int count = 0;
    // Length of all three strings
    int l1 = s1.length();
    int l2 = s2.length();
    int l3 = s3.length();
    // vector to store frequency of each char
    vector<int> freq(26);
    for (int i = 0; i < l1; i++)
        ++freq[s1[i] - 'A'];
    for (int i = 0; i < l2; i++)
        ++freq[s2[i] - 'A'];
    for (int i = 0; i < l3; i++)
        ++freq[s3[i] - 'A'];
    // count number of unique char
    for (int i = 0; i < 26; i++)
        if (freq[i] > 0)
            count++;
    // solution not possible for count greater than 10
    if (count > 10)
    {
        cout << "ERROR";
        return 0;
    }
    // array of nodes
    node nodeArr[count];
    // store all unique char in nodeArr
    for (int i = 0, j = 0; i < 26; i++)
    {
        if (freq[i] > 0)
        {
            nodeArr[j].c = char(i + 'A');
            j++;
        }
    }
    bool sol= permutation(count, nodeArr, 0, s1, s2, s3);
    if (sol)
    {
    string output;
    for (int i=0; i<s1.length();i++)
    {
            char letter=s1[i];
            for (int j = 0; j < count; j++)
            {
                    if (nodeArr[j].c==letter)
output.append(to_string(nodeArr[j].v));
            }
    }
    output.append("+");
    for (int i=0; i<s2.length();i++)
    {
            char letter=s2[i];
            for (int j = 0; j < count; j++)
            {
                    if (nodeArr[j].c==letter)
output.append(to_string(nodeArr[j].v));
            }
    }
    output.append("=");
    for (int i=0; i<s3.length();i++)
    {
            char letter=s3[i];
            for (int j = 0; j < count; j++)
            {
                    if (nodeArr[j].c==letter)
output.append(to_string(nodeArr[j].v));
            }
    }
    cout << output;
    }
    return sol;
}
int main()
{
    string s1, s2, s3;
    cin>>s1;
    cin>>s2;
    cin>>s3;
    if (solveCryptographic(s1, s2, s3) == false)
        cout << "NO SOLUTION";
    return 0;
}
```

30 The trapped knight

20 points

Introduction

The "Trapped Knight" is a chess-based problem invented by Neil Sloane. It revolves around an infinite chess board with spiraling numbers set on it as seen on the following image, where we put a knight in the center:

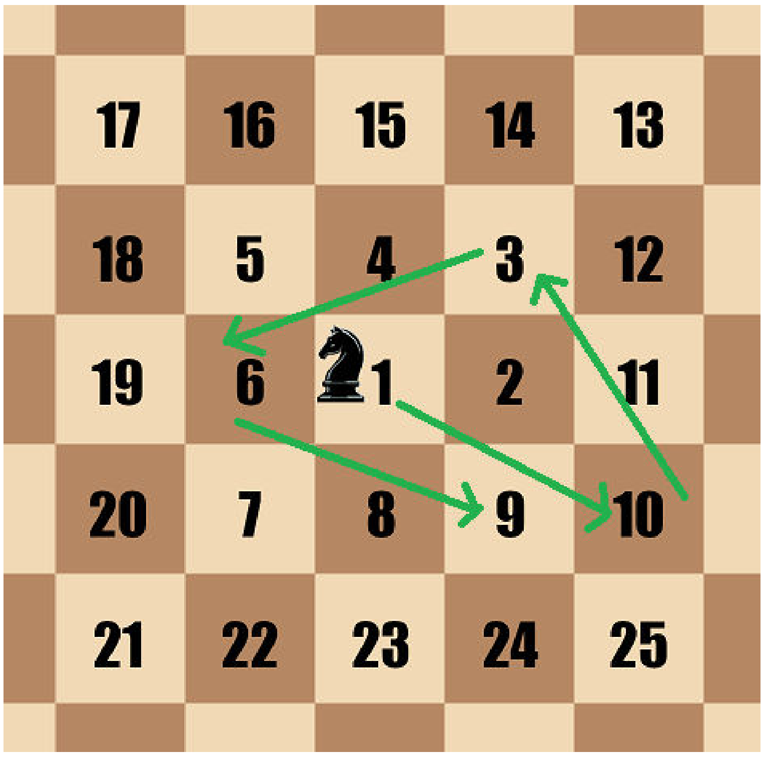


From that square, the knight moves with these rules:

1. Its movement is the usual chess knight movement.

1. From all the potential squares it can jump to, it will pick the square with the lowest number as long as it has not visited it before.

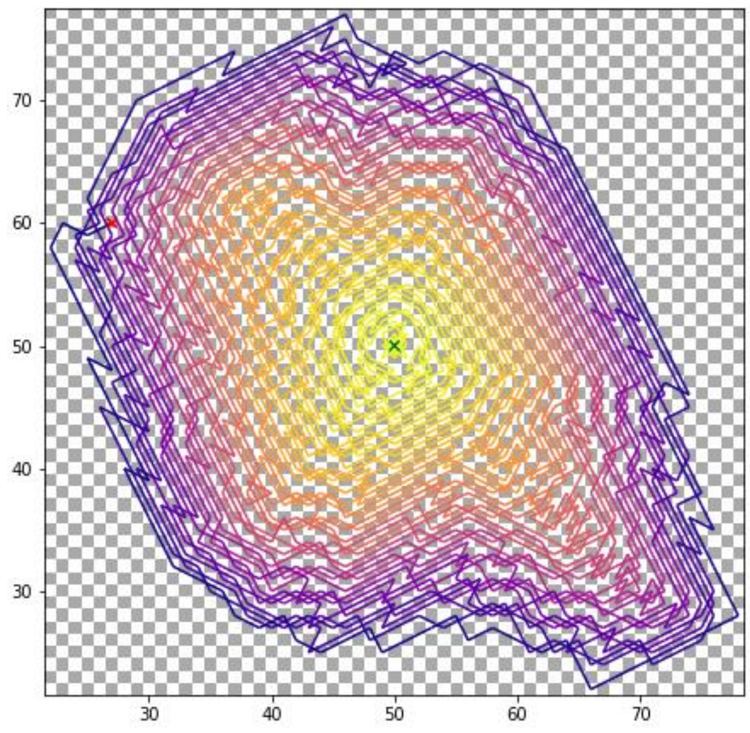
Example:



This creates a sequence of visited squares:

```
1, 10, 3, 6, 9, 4, 7, 2...
```

It is know that after a determinate number of jumps, the knight ends trapped in the square with number 2084, where it can't jump to any non-visited square anymore. You can see a diagram of its jumps in the following image:



Exercise

Create a program that, given the number of jumps of the knight, returns the value of the square where it's sitting. In case the knight is already trapped, ignore all the extra jumps after the last one and return the value of the square where it's trapped.

Note: a 75x75 grid is enough to capture all the knight's movements before it gets trapped.

Input

A positive number: the number of movements for the knight.

Output

Another positive number: the value of the square where the knight is placed.

Example 1

Input

```
0
```

Output

```
1
```

Example 2

Input

```
1
```

Output

```
10
```

Example 3

Input

```
4
```

Output

```
9
```

Example 4

Input

```
2014
```

Output

```
2467
```

Example 5

Input

```
2015
```

Output

```
2084
```

Example 6

Input

```
3000
```

Output

```
2084
```

Solutions

Python

```
import sys
```

```
def create_grid():
    grid_size = 75  # this has to be odd so the first number is in the center
    grid = [[0]*grid_size for i in range(grid_size)]  # create empty grid
    ix = iy = grid_size // 2  # initial index at the center of the grid
    # order of east, north, west, south
    directions = [[1, 0], [0, -1], [-1, 0], [0, 1]]
    leaps = [1, 1, 2, 2]  # number of leaps to do in every direction
    # initialize to first and
    directions_idx = 0
    leap_idx = 0
    leaps_pending = leaps[0]
    for i in range(grid_size*grid_size):
        grid[iy][ix] = i+1
        if leaps_pending > 0:
            leaps_pending -= 1
        else:
            directions_idx += 1
            leap_idx += 1
            if directions_idx > 3:
                # increase leaps by two, the spyral grows two squares in
every direction
                leaps = [x+2 for x in leaps]
                directions_idx = 0
                leap_idx = 0
            # when changing direction we jumping right away, substract one to
the leaps
            leaps_pending = leaps[leap_idx]-1
        ix += directions[directions_idx][0]
        iy += directions[directions_idx][1]
    return grid
def do_jumps(grid, n_jumps):
    ix = iy = len(grid) // 2  # start at center of grid
    jumps = [[-1, -2], [-1, 2], [1, -2], [1, 2],
             [-2, -1], [-2, 1], [2, -1], [2, 1]]  # knight moves
    visited = [(ix, iy)]  # add where we start
    result = grid[iy][ix]
    for _ in range(n_jumps):
        min = None
        jump_x = jump_y = 0
        for jump in jumps:
            target_x, target_y = (ix + jump[0], iy + jump[1])
            if (target_x, target_y) not in visited:
                value = grid[target_y][target_x]
                if min is None or value < min:
                    min = value
                    jump_x, jump_y = (target_x, target_y)
        if min is not None:
            result = min
            ix, iy = (jump_x, jump_y)
            visited.append((ix, iy))
        else:
            # min not found, that means the knight can't jump anymore
            break
    return result
def knight(n_jumps):
    grid = create_grid()
    result = do_jumps(grid, n_jumps)
    print(result)
def main():
    n_jumps = int(input())
    knight(n_jumps)
if __name__ == "__main__":
    main()
```

C++

```
#include <iostream>
#include <set>
const int GridSize = 75;
int main()
{
    int nJumps;
    std::cin >> nJumps;
    int grid[GridSize][GridSize];
    // empty grid, not actually needed
    for (auto& i : grid)
        for (int& j : i)
            j = 0;
    int ix;
    int iy = ix = GridSize / 2; // start at the center
    const int directions[4][2] = { {1, 0}, {0, -1}, {-1, 0}, {0, 1} }; //
east, north, west, south
    int leaps[4] = { 1, 1, 2, 2 };
    int directionsIdx = 0;
    int leapIdx = 0;
    int leapsPending = leaps[0];
    // grid creation
    for (int i = 0; i < GridSize*GridSize; i++)
    {
        grid[iy][ix] = i + 1;
        if (leapsPending > 0)
            leapsPending -= 1;
        else
        {
            directionsIdx++;
            leapIdx++;
            if (directionsIdx > 3)
            {
                // increase leaps by two, the spyral grows two squares in
every direction
                for (int& leap : leaps)
                    leap += 2;
                directionsIdx = 0;
                leapIdx = 0;
            }
            // when changing direction we jumping right away, substract one
to the leaps
            leapsPending = leaps[leapIdx] - 1;
        }
        ix += directions[directionsIdx][0];
        iy += directions[directionsIdx][1];
    }
    // reset to center
    ix = iy = GridSize / 2;
    const int jumps[8][2] = { {-1, -2}, {-1, 2},{1, -2}, {1, 2}, {-2, -1}, {-
2, 1}, {2, -1}, {2, 1} }; // knight moves
    std::set<std::pair<int,int>> visited;
    visited.insert({ ix, iy });
    int result = grid[iy][ix];
    for (int i = 0; i < nJumps; i++)
    {
        int min = 0;
        int jumpX;
        int jumpY = jumpX = 0;
        for (auto jump : jumps)
        {
            int targetX = ix + jump[0];
            int targetY = iy + jump[1];
            if (visited.find({targetX, targetY}) == visited.end())
            {
                const int value = grid[targetY][targetX];
                if (min == 0 || value < min)
                {
                    min = value;
                    jumpX = targetX;
                    jumpY = targetY;
                }
            }
        }
        if (min > 0)
        {
            result = min;
            ix = jumpX;
            iy = jumpY;
            visited.insert({ ix, iy });
        }
        else
            // min not found, that means the knight can't jump anymore
            break;
    }
    std::cout << result << std::endl;
}
```