CODEWARS | hp

SPAIN **Virtual** Edition 2021

Problems

And

Solutions

#	Problem	Points
1	How many submissions	1
2	Grace Hopper	2
3	Printing Plan Calculator	2
4	Building triangles	3
5	Margarita Salas	3
6	Crochet planning	4
7	Dermatoglyphics	4
8	Simplest sudoku	4
9	Civilization group	5
10	Fantasy Trainer TM	5
11	Segapotoa blacklist	5
12	Katie Bouman	6
13	The Student Grade Calculator	6
14	The Battle Royal Game	9
15	The Barnum-Forer Horoscope	9
16	Skin Cancer	9
17	Periodic numbers	10
18	Fischer Clock	10
19	Image Scale	10
20	Task Manager	11
21	Toc-Bum	11
22	Automatic Tuner	12
23	Mario Prize	12
24	Flower classification	12
25	Back To The Future	13
26	Autocomplete	13
27	Tactile Writing	15
28	Dilahk's contest	17
29	Adaptive Mean Thresholding	21
30	Viking Assault	24
31	Pharaoh Puzzle	25
32	Curie's Waterfall	35

1 How many submissions

1 points

Introduction

George and his friends enjoy programming contests like HP CodeWars. So, they created their own contest to train and have fun coding with friends.

As you can imagine, the contest server must be a robust component to manage submissions that can potentially freeze the system. George found out that the hard disk space is not released after each submission, so they need to calculate the space they will need according the amount of submissions expected.

Let's help them with a simple program that does the math for us.

Input

The input will be a pair of positive integer values.

The first value is the total amount of submissions made during the contest.

The second value in MB (megabytes) represents the memory needed to evaluate a submission.

Output

The output will be a positive integer value, also in MB, representing the total space needed to run the contest comfortably.

Example

Input

```
522
980
```

Output

```
511560
```

Python3

```
diskSize = int(input())
virtualEnvironment = int(input())

print(diskSize*virtualEnvironment)
```

Java

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args) throws Exception {
        Scanner reader=new Scanner(System.in);  // Read from standard input (req
uired for the Judge)
        int submissions = reader.nextInt();
        int submissionMemory = reader.nextInt();
        int totalMemory = submissionMemory *submissions;
        System.out.println(String.valueOf(totalMemory));
        reader.close();
    }

}
```

2 Grace Hopper

2 points

Introduction

Grace Hopper (1906-1992) was a computer pioneer and naval officer. She was a pioneer in the world of computer science and one of the first programmers of the Harvard Mark I computer. She popularized the idea of machine-independent programming languages, which led to the development of COBOL, an early high-level programming language still in use today.

Hopper tried to enlist in the US Navy during World War II but had to join the army's reserve because she was already 34 years old.

The simplest program in any programming language is the popular printing of "Hello word!" message. Here you have the version for COBOL language:

```
IDENTIFICATION DIVISION.
PROGRAM-ID. HELLO-WORLD.
* simple hello world program
PROCEDURE DIVISION.
    DISPLAY 'Hello world!'.
    STOP RUN.
```

HINT: be careful, indentations should be printed as four whitespace characters.

We would like you to program a simple program that outputs the COBOL program to printout of a given string.

Input

The input will be a single line containing the message to print.

Output

The output is the COBOL program needed to print the message received in the input.

Example

Input

```
Hi Grace!
```

Output

```
IDENTIFICATION DIVISION.
PROGRAM-ID. HELLO-WORLD.
* simple hello world program
PROCEDURE DIVISION.
    DISPLAY 'Hi Grace!'.
    STOP RUN.
```

Python3

```
display = str(input())

template = "IDENTIFICATION DIVISION.\nPROGRAM-ID. HELLO-WORLD.\n" \
           "* simple hello world program\nPROCEDURE DIVISION.\n" \
           "    DISPLAY '%s'.\n    STOP RUN." % display
print(template)
```

Java

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args) throws Exception {
        Scanner reader=new Scanner(System.in);  // Read from standard input (req
uired for the Judge)
        String message = reader.nextLine();
        System.out.println(String.format("IDENTIFICATION DIVISION.\nPROGRAM-
ID. HELLO-
WORLD.\n* simple hello world program\nPROCEDURE DIVISION.\n    DISPLAY '%s'.\n
  STOP RUN.", message ));
        reader.close();
    }

}
```

3 Printing Plan Calculator

2 points

Introduction

HP has a revolutionary printing program: HP Instant Ink. With this program, the user can pay only for the number of copies printed and not for the volume of ink consumed.

HP Instant Ink has 5 plans:

Plan Name	Price (Euros)	Number of pages printed per month
Free	0	15
Occasional	2.99	50
Moderate	4.99	100
Frequent	9.99	300
Business	19.99	700
Not available	-	>700

Take into account that the number of pages printed per month is the maximum pages allowed for a specific plan (value itself included).

Input

The input value will be a positive integer value representing the number of pages that user prints per month.

Output

The output will be the name of the best plan for that user and the price per month that the user must pay.

Example 1

Input

```
15
```

Output

```
Plan in order to print 15 pages per month:
Free.
Price: 0 Euros.
```

Example 2

Input

```
450
```

Output

```
Plan in order to print 450 pages per month:
Business.
Price: 19.99 Euros.
```

Example 3

Input

```
1400
```

Output

```
Plan in order to print 1400 pages per month:
Not available.
Price: - Euros.
```

Python3

```python
pages = int(input())

print("Plan in order to print %s pages per month:" % pages)
if pages <= 15:
    print("Free.")
    print("Price: 0 Euros.")
elif pages <= 50:
    print("Occasional.")
    print("Price: 2.99 Euros.")
elif pages <= 100:
    print("Moderate.")
    print("Price: 4.99 Euros.")
elif pages<=300:
    print("Frequent.")
    print("Price: 9.99 Euros.")
elif pages <= 700:
    print("Business.")
    print("Price: 19.99 Euros.")
else:
    print("Not available.")
    print("Price: - Euros.")
```

C++

```
#include <iostream>

int main()
{
    // Read pages printed
    int pages = 0;
    std::cin >> pages;

    // Check which is the best plan
    std::string plan = "";
    std::string price = "";
    if ( pages <= 15)
    {
        plan = "Free";
        price = "0";
    }
    else if ( pages <= 50 )
    {
        plan = "Occasional";
        price = "2.99";
    }
    else if ( pages <= 100 )
    {
        plan = "Moderate";
        price = "4.99";
    }
    else if ( pages <= 300 )
    {
        plan = "Frequent";
        price = "9.99";
    }
    else if ( pages <= 700 )
    {
        plan = "Business";
        price = "19.99";
    }
    else
    {
        plan = "Not available";
        price = "-";
    }

    // Print result
    std::cout << "Plan in order to print " << pages << " pages per month:" << st
d::endl;
```

```
    std::cout << plan << "." << std::endl;
    std::cout << "Price: " << price << " Euros." << std::endl;

    return 0;
}
```

4 Building triangles

3 points

Introduction

You get a summer job in a triangle factory as a quality assurance manager. Your duty is to verify if a triangle can be built given three sides. To ease your job, you remember the Triangle Inequality Theorem and decide to write a simple program to perform this check.

Well, just in case you do not remember this theorem here you have it: The Triangle Inequality Theorem states that the sum of any two sides of a triangle, must be greater than the length of the third side of a triangle.

Input

The input will be three different lines, one corresponding to each side of the triangle and containing a positive integer value. Assume that the three sides will be expressed with the same units.

Output

The output will print a single line. In case that the three sides can be used for building a triangle:

```
It is a triangle
```

Otherwise the output will be:

```
It is NOT a triangle
```

Example 1

Input

```
600
900
700
```

Output

```
It is a triangle
```

Example 2

Input

```
600
900
300
```

Output

```
It is NOT a triangle
```

Python3

```python
a = int(input())
b = int(input())
c = int(input())
if (a + b > c) and (a + c > b) and (b + c > a):
  print("It is a triangle")
else:
  print("It is NOT a triangle")
```

5 Margarita Salas

3 points

Introduction

Margaritas Salas (1938-2019) was a Spanish renowned scientist in the fields of biochemistry and molecular genetics.

Disciple of Severo Ochoa, with whom she worked in the United States, she invented a faster, simpler and more reliable way to replicate trace amounts of DNA into quantities large enough for full genomic testing. Her invention based on Phi-29 DNA polymerase is now used widely in oncology, forensics and archaeology.

As you may know, the DNA is formed by the sequence of four bases: adenine (A), guanine (G), cytosine (C), and thymine (T). Given a DNA sequence, write down a program to replicate it as many times as requested.

Input

The input will be a pair of lines.

The first line contains the number of copies, bigger than zero, to replicate the DNA sequence.

The second line represents the DNA sequence that must be replicated.

Output

The output is the DNA sequence replicated as many times as requested.

Example 1

Input

1

ACGT

Output

ACGT

Example 2

Input

3

GATTACA

Output

GATTACAGATTACAGATTACA

Python3

```
num = int(input())
dnaChain =input()

replicatedChain = ""
for i in list(range(num)):
  replicatedChain += dnaChain

print(replicatedChain)
```

6 Crochet planning

4 points

Introduction

During this winter, granny Mary has spent a good amount of time next to her fireplace doing crochet while watching tv and the pictures of her grandchildren. Consequently, her production has been quite high. Now she is planning an improvement for next winter: to precisely predict the amount of yarn needed (since it has always been a problem). Although she worked as a programmer in the old days, quite some time has passed and a lot has changed, so she is asking you to help her by creating a program capable of carrying it out.

The croquet technique is based on the premise that the consumption of yarn depends on the type of stich. Mary has defined the following:

- Chain, represented by 'o', consumes 1cm
- Single crochet, represented by '+', consumes 1.5cm
- Double crochet, represented by 'T', consumes 2cm

Input

Single or multiple lines containing the characters granny Mary used to represent each type of stich. When reading a '#' it will mean the end of the sequence.

Output

Amount of yarn required in cm.

Example 1

Input

```
oT++T
T+ooo
#
```

Output

```
14.5
```

Example 2

Input

```
o+T
#
```

Output

```
4.5
```

Python3

```
import sys

total = 0
input_str = sys.stdin.read()

for c in input_str:
    if c == 'o':
        total += 1
    elif c == '+':
        total += 1.5
    elif c == 'T':
        total += 2

print(total)
```

C++

```
#include <iostream>

using namespace std;

int main()
{
    char  aStich;
    float aLength = 0;

    //input data
    while( cin >> aStich)
    {
        switch( aStich ) {
            case 'o' : aLength += 1; break;
            case '+' : aLength += 1.5; break;
            case 'T' : aLength += 2; break;
            default: break;
        }
    }

    cout << aLength;
}
```

7 Dermatoglyphics

4 points

Introduction

Dermatoglyphics, what a word! It is the science of the study of skin patterns observed in fingerprints, lines, mounts and shapes of hands. But this word is special for another reason, it is the longest isogrammic word known in English. An isogrammic word, is a word without a repeating letter.

Checking whether a word is an isogrammic one or not can be so tedious. Can you write a simple program to determine whether a word is isogrammic or not?

Input

The input will be a single word.

Output

The output will print a single line stating if the word is isogrammic:

```
Isogram detected
```

Otherwise the output will be:

```
Not an isogram
```

Example 1

Input

```
pool
```

Output

```
Not an isogram
```

Example 2

Input

```
Dermatoglyphics
```

Output

```
Isogram detected
```

Python3

```python
word = list(input().lower())
isogram = True

while len(word)>1:
    letter = word.pop(0)
    if letter in word:
        isogram = False
        break

if isogram:
    print("Isogram detected")

else:
    print("Not an isogram")
```

8 Simplest sudoku

4 points

Introduction

A sudoku is a japanese puzzle in which players insert the numbers 1 to 9 into a grid, consisting of nine squares, which are further subdivided into nine smaller squares, in such a way that every number appears once in each horizontal line, vertical line, and square.

To make things easier, we are going to play with one of the nine smaller squares. We would like you to develop a simple program, that checks that the smaller square has all the numbers from 1 to 9 only once. In case that the square has repeated numbers, we would like to know which are the ones that are missing.

Input

The input will be always three lines. Each line containing three numbers from 1 to 9 separated by white spaces.

Output

The output will print a single line when the square is valid:

```
This is a valid sudoku
```

Otherwise the output will print a different single line followed by a variable number of lines containing the missing numbers (one number per line) ordered:

```
This is an invalid sudoku. Missing numbers are:
```

Example 1

Input

```
1 2 3
4 5 6
7 8 9
```

Output

```
This is a valid sudoku
```

Example 2

Input

```
1 1 2

3 4 5

5 6 7
```

Output

```
This is an invalid sudoku. Missing numbers are:

8

9
```

Python3

```python
numbers = [int(x) for x in input().split(' ')]
numbers += [int(x) for x in input().split(' ')]
numbers += [int(x) for x in input().split(' ')]

missing = []
for n in range(1, 10):
    if n not in numbers:
        missing.append(n)

if not missing:
    print("This is a valid sudoku")
else:
    print("This is an invalid sudoku. Missing numbers are:")
    for n in missing:
        print(n)
```

9 Civilization group

5 points

Introduction

An ancient civilization built an infinite odd numbers pyramid, with the purpose of knowing the civilization-group of each member. They wrote it down on to a stone.

The shaman was the first and his civilization-group was the 1, his family was the third and the fifth and they belonged to the civilization-group 8. This way, they could classify people in civilization-groups. The civilization-group is the sum of all the numbers that are in a row, so for 1 is 1, for 3 & 5 is 8, for 7 & 9 & 11 is 27, and so on...

```
1:          1                       --> civilization-group: 1
2:        3   5                     --> civilization-group: 8
3:     7    9   11                  --> civilization-group: 27
4:   13   15   17   19              --> civilization-group: 64
           ...
```

HINT: do not try to hardcode all the possible solutions

Input

The input to the problem will be a number that indicates the row number.

Output

Should be the sum of all the odd numbers of the row indicated in the input.

Example 1

Input

2

Output

8

Example 2

Input

12

Output

1728

Python3

```python
index = int(input())
print(index*index*index)
```

10 Fantasy Trainer TM

5 points

Introduction

Fantasy Trainer TM is a popular fantasy-like football game that can be played using a mobile app.

In the game, each participant plays with a group of friends and has to build a team based on real-life football players, where each player is rewarded points based on statistics and goals, with the aim to have their team finish first and to win honor and glory. Or at least not to be last, which usually involves paying for refreshments and chocolate to the rest of the group.

To help us beat our friends, we want to develop a simple program that will read a set of real-life players with the points they earned in the latest game and will return which of them is the MVP (Most Valuable Player).

You must note that the score system consists of two different parts:

- Spades: Based on statistics like the percentage of good passes or the fouls, each spade earns 2 points.
- Goals: Each goal will directly earn 4 points

Input

The input will be divided in two parts:

First, a positive number "N" equal or greater than two that will tell us the number of players to consider.

Secondly, "N" lines, each one for a player, in the format: "PLAYER scored G goal(s) and earned S spade(s)" where G and S are both positive numbers, including zero.

Output

The name of the player with most points, followed by the score:

```
PLAYER is the MVP with P points!
```

If more than one can be selected, the program will just return the code word:

```
DRAW
```

Example 1

Input

```
2
Roger scored 1 goal(s) and earned 3 spade(s)
David scored 0 goal(s) and earned 1 spade(s)
```

Output

```
Roger is the MVP with 10 points!
```

Example 2

Input

```
3
Abel scored 1 goal(s) and earned 5 spade(s)
Adur scored 0 goal(s) and earned 4 spade(s)
Ana scored 2 goal(s) and earned 3 spade(s)
```

Output

```
DRAW
```

C++

```cpp
#include <stdint.h>
#include <iostream>
#include <stdlib.h>
#include <cmath>
#include <vector>
#include <set>

std::vector<std::string> names_ = std::vector<std::string>();
std::vector<int> scores_ = std::vector<int>();

int GetWinner()
{
    int winner = 0;
    bool draw = false;
    for (int iCnt = 1; iCnt < scores_.size(); iCnt++)
    {
        if (scores_[iCnt] > scores_[winner] )
        {
            winner = iCnt;
            draw = false;
        }
        else if (scores_[iCnt] == scores_[winner] )
        {
                draw = true;
            }
        }

        return (draw?-1:winner);
}

int main()
{
    std::string sAux;

    // Read number of players
    int playerNumber;
    std::cin >> sAux;
    playerNumber = atoi(sAux.c_str());
    // Read statements
    for (int iCnt = 0; iCnt < playerNumber; iCnt++)
    {
        // The line is
        // <Name> scored X goal(s) and earned Y spade(s)
        for (int jCnt = 0; jCnt < 8; jCnt++)
        {
            std::cin >> sAux;
            if (jCnt == 0) // Name
            {
                names_.push_back(sAux);
            }
            else if (jCnt == 2) // goals
            {
                scores_.push_back(atoi(sAux.c_str()) * 4);
            }
            else if (jCnt == 6) // spades
            {
                scores_[scores_.size()-1] += atoi(sAux.c_str()) * 2;
            }
        }
    }

    int winner = GetWinner();
    if (winner == -1)
    {
        std::cout << "DRAW\n";
    }
    else
    {
        std::cout << names_[winner] << " is the MVP with " << scores_[winner] <<
 " points!\n";
```

```
    }
}
```

Python3

```
n = int(input())

players = dict()
for i in range(n):
  words = input().split()
  name = words[0]
  goals = int(words[2])
  spades = int(words[6])
  points = goals * 4 + spades * 2
  players[name] = points

maximum = max(players.values())
bests = [k for k,v in players.items() if v == maximum]

if len(bests) == 1:
  mvp = bests[0]
  print(mvp + ' is the MVP with ' + str(players[mvp]) + ' points!')
else:
  print('DRAW')
```

11 Segapotoa blacklist

5 points

Introduction

Segapotoa is a new operating system for smartphones, which is trying to disrupt the market. As a new employee you have just been handed the problem of the spam calls, which for sure will bother the users of Segapotoa. In order to avoid that, a new functionality has to be developed. Thus, it is necessary to compare the incoming calls with a defined blacklist in order to know if it is a spam call.

Input

The first input of the program is a predefined list of blacklisted phone numbers separated in different lines and finishing with a 0. The second input is the number of the incoming call. The phone number will follow the Spanish nomenclature, that is to say, only digits will be allowed and at most 9 of them.

Output

The output will be 1 if the incoming call is in the blacklist and 0 otherwise.

Example 1

Input

```
937116189
689303064
943282927
0
1004
```

Output

```
0
```

Example 2

Input

```
937116189
689303064
943282927
112
0
689303064
```

Output

```
1
```

C++

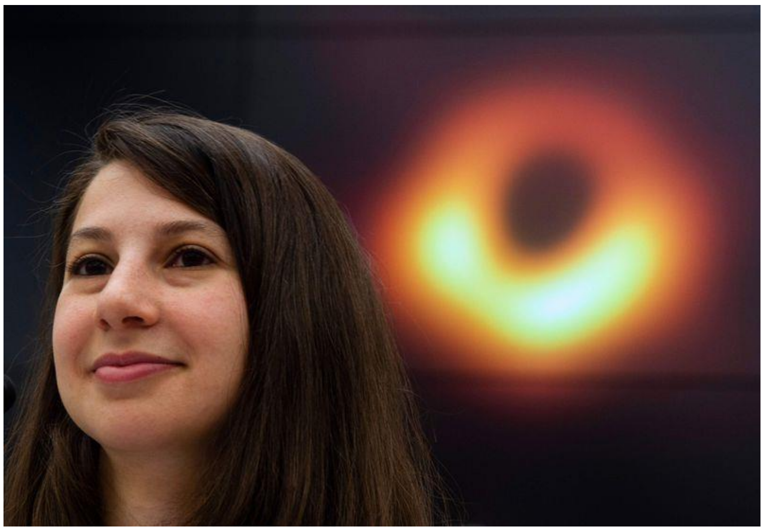
```cpp
#include <iostream>
#include <vector>

using namespace std;

int main()
{
    int result = 0;
    int call;
    vector<int> blacklist;
    int incomingCall;

    cin >> call;
    while (call != 0){
        blacklist.push_back(call);
        cin >> call;
    }

    cin >> incomingCall;

    for(const auto &i : blacklist)
    {
        if(i == incomingCall)
        {
            result = 1;
            break;
        }
    }

    cout << result << endl;
}
```

12 Katie Bouman

6 points

Introduction

Katie Bouman (1989) is a PhD student in computer science and artificial intelligence at the Massachusetts Institute of Technology (MIT). Four years ago, she led the creation of an algorithm that would eventually lead to the first image of a supermassive black hole at the heart of the Messier 87 galaxy, some 55 million light years away from Earth.



The data used to piece together the image was captured by the Event Horizon Telescope (EHT), a network of eight radio telescopes spanning from Antarctica to Spain and Chile. Bouman's role, when she joined the team working on the project seven years ago as a 23-year-old junior researcher, was to help build an algorithm which could construct the masses of astronomical data collected by the telescope into a single coherent image.

As an example, consider you get 4 partial images of 2 columns and 3 rows like these:

```
00   11   11   00
01   00   00   10
00 , 11 , 11 , 00
```

Then you can concatenate them creating the final image:

```
00   11   11   00   00111100
01 + 00 + 00 + 10 = 01000010
00   11   11   00   00111100
```

Just to make you feel like Katie, could you write a program that constructs an image from partial small images?

Input

The first line of the input contains the number of columns and rows of each partial small image.

The second line represents the total amount of partial small images to be joined sequentially.

Then the following lines will represent each of the small images.

Output

The output is the final image formed by the concatenation of the partial small images.

Example 1

Input

```
2 3
4
00
01
00
11
00
11
11
00
11
00
10
00
```

Output

```
00111100
01000010
00111100
```

Example 2

Input

```
2 2
5
XX
XX
YY
YY
ZZ
ZZ
00
00
11
11
```

Output

```
XXYYZZ0011
XXYYZZ0011
```

Python3

```
cols, rows = input().split()
cols = int(cols)
rows = int(rows)
numPartialImages = int(input())

# create the matrix of #rows to store the input data
l = []
for j in range(rows):
  l.append([])

# read the input and store it by rows forming the final image
for i in range(numPartialImages):
  for j in range(rows):
    l[j].append(input())

# print the image line by line
for j in range(rows):
  print("%s" % "".join(l[j]))
```

13 The Student Grade Calculator

6 points

Introduction

Thyra is a Science teacher, and during this quarter she has delivered homework and several exams to her students.

She needs to compute the final grade for each student, considering that not all exams nor homework assignments have the same weight regarding the final grade...

Help Thyra write a program that computes the weighted arithmetic mean given the grades and weights per each student. The weighted arithmetic mean is computed with this formula:

$$\overline{x}_w = \sum_{i=1}^{N} x_i \cdot w_i$$

Where *xi* are the grades and *wi* are the corresponding weights. In case you don't remember what the uppercase sigma means, in mathematics it represents a summation: the addition of a sequence of any kind of numbers. In this case it is the sum of the product *xi·wi* for each value of *i*.

Input

Input must be two lines, the first containing the student school grades and the second the weight of each grade.

Values will be separated by one space. Grades are numbers between 0.0 and 10.0. Weights are numbers between 0.0 and 1.0.

Output

The output will print a single line with the weighted arithmetic mean of the input values considering the weights. Only output one decimal digit.

`The student final grade is` *weighted_arithmetic_mean_value*

Example 1

Input

```
6.5 7.7 7.2 10.0 7.0 8.5
0.1 0.1 0.1 0.1 0.2 0.4
```

Output

```
The student final grade is 7.9
```

Example 2

Input

```
8.5 9.0 10.0 8.6
0.1 0.2 0.2 0.5
```

Output

```
The student final grade is 8.9
```

C++

```cpp
#include <iostream>
#include <vector>
#include <iomanip>

int main ()
{
    std::vector<double> grades;
    std::vector<double> weights;
    double value;

    while (std::cin.peek() != '\n')
    {
        std::cin >> value;
        grades.push_back(value);
    }

    for (int i = 0; i < grades.size(); ++i)
    {
        std::cin >> value;
        weights.push_back(value);
    }

    value = 0;
    for (int i = 0; i < grades.size(); ++i)
    {
        value += grades[i]*weights[i];
    }

    std::cout << std::fixed << std::setprecision(1) << "The student final grade
is " << value << std::endl;
}
```

Python3

```python
grades = input().split(" ")
weights = input().split(" ")

finalGrade = 0

for i in range(len(grades)):
    finalGrade = finalGrade + float(grades[i]) * float(weights[i])

print(f"The student final grade is {finalGrade:.1f}")
```

14 The Battle Royal Game

9 points

Introduction

You work in the development of a new battle royal game.

The players will fight each other inside a fixed map size, but every 10 minutes the size of the map gets reduced and any player outside the limits will lose the game.

You have been requested to develop a small prototype to check how the map limits will work.

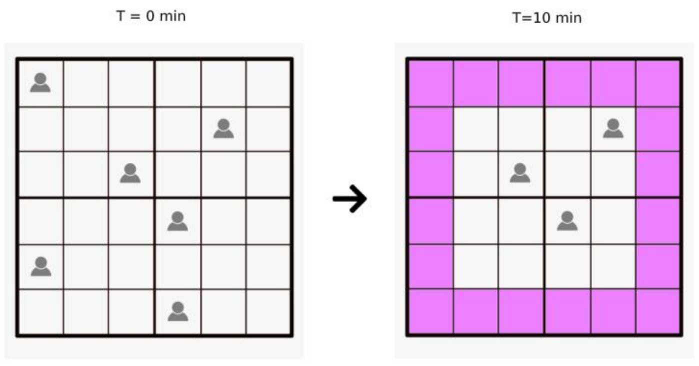
HINT:

The map is defined NxN

The valid player's positions go from 0 to N-1

Note that the map gets reduced by all sides! (top, bottom, right, left) and that players do not move from their positions

Here's an example.



Input

The input will be:

- The size of the map given as maximum rows and columns (only squares are valid 3x3,4x4,5x5 etc).

- The match time in minutes in which we want to evaluate the number of players remaining.
- Several positions on the map, representing the player's position at the beginning of the match. (T=0 min)

The process should stop reading player's position when the program finds the character '#'.

```
4 4
15
1 2
3 3
#
```

In this example the first line describes that this is a 4x4 map.

The second line refers to 15 minutes from the start of the match.

The third line provides the position of a player on row 1 and column 2.

At the fourth line another player position is set on row 3 and column 3.

Finally the character '#' marks stop reading input file.

Output

The output will be the number of players that remain in the match at that specific point in time.

Example 1

Input

```
4 4
15
1 2
3 3
#
```

Output

```
1
```

Example 2

Input

```
8 8
25
3 4
2 1
5 5
#
```

Output

```
2
```

Python3

```
size = ([int(coord) for coord in input().split()])
time = int(input())
jugadors = []
while True:
    jug = input()  # .split()
    if str(jug) != "#":
        jugadors.append([int(coord) for coord in jug.split()])
    else:
        break

iterations = int(time) // 10
alive = []
for jugador in jugadors:
    if not ((((jugador[0]) < iterations) or (jugador[0] > size[0]-
1 - iterations)) or
            ((jugador[1] < iterations) or (jugador[1] > size[1]-
1 - iterations))):
        alive.append(jugador)
print(len(alive))
```

C++

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

using namespace std;

int main(int argc, char *argv[])
{
    //Read information
    int numRows = 0;
    int numCols = 0;
    int elapsedTime = 0;
    cin >> numRows;
    cin >> numCols;
    cin >> elapsedTime;

    bool map[numRows][numCols];
    for(int i = 0; i < numRows; i++)
        for(int j = 0; j < numCols; j++)
            map[i][j] = false;

    //Number of 10 min intervals
    int elapsedIntevals = 0;
    elapsedIntevals = elapsedTime / 10;
    int playerPositionX = -1;
    int playerPositionY = -1;

    string auxReaded;
    cin >> auxReaded;
    while(auxReaded != "#")
    {
        int auxIntRead = std::stoi(auxReaded);

        if(playerPositionX == -1)
            playerPositionX = auxIntRead;
        else
        {
            playerPositionY = auxIntRead;
            map[playerPositionX][playerPositionY] = true;

            playerPositionX = playerPositionY = -1;
        }
       cin >> auxReaded;
    }

    //Check player with life
    int playersAlife = 0;
    for(int i= 0 + elapsedIntevals ; i < numRows - elapsedIntevals; i++)
    {
        for(int j= 0 + elapsedIntevals ; j < numCols - elapsedIntevals; j++)
        {
            if(map[i][j])
                playersAlife++;
        }
    }
    cout<<playersAlife;

    return 0;
}
```

15 The Barnum-Forer Horoscope

9 points

Introduction

You are a group of psychological researchers and you are designing an experiment to verify the existence of the Barnum-Forer effect via an a experiment.

The Barnum-Forer effect says that when there is a set of generic sentences presented to individuals, they feel that the sentence fits perfectly to his or her personality and has been redacted specifically for him/her. To conduct this experiment you will provide sentences to each participant as a kind of "horoscope", but instead of using just the date of birth you will consider the name of the subject as well.

To implement the Barnum-Forer Horoscope we will first take the date of birth in format DDMMYY. For instance, February 9th of 2019 will be represented as the string "090219". To this number, we will add all the individual digits: 0+9+0+2+1+9. The result of the operation 21 will be processed again to add all the digits. The algorithm will keep adding all the digits until only a single digit is returned. In this case 2+1=3 so the final result of the operation with the date of birth in this example would be 3. Then we will take the name of the subject and will convert each character into its ASCII code, sum all the values and proceed as before. Adding all the numbers repeatedly until we obtain a single number.

For instance, if the name is "Sofia" we will convert each character to its ASCII code:

S -> 83, o -> 111, f -> 102, i -> 105, a -> 97. We will sum 83 + 111 + 102 + 105 + 97 = 498. Then 4+9+8 = 21. And finally, 2 + 1 = 3.

To finish with the algorithm, we will take the first number that we obtained from the date of birth and the digit computed from the name and concatenate both.

In this example the result of concatenating both numbers is "33". We then keep doing the same sum of digits until we finish with a single-digit: 3+3 = 6

Six is the final number that we will use to look into a table of strings. This string is the output that we will give to the subjects of the experiment.

The table with each number and the associated sentence is the following:

- 1 -> "You tend to be critical of yourself."

- 2 -> "You have a great deal of unused capacity which you have not turned to your advantage."
- 3 -> "While you have some personality weaknesses, you are generally able to compensate for them."
- 4 -> "At times you are extroverted, affable, sociable, while at other times you are introverted, wary, reserved."
- 5 -> "Disciplined and self-controlled outside, you tend to be worrisome and insecure inside."
- 6 -> "At times you have serious doubts as to whether you have made the right decision or done the right thing."
- 7 -> "You prefer a certain amount of change and variety and become dissatisfied when hemmed in by restrictions and limitations."
- 8 -> "You pride yourself as an independent thinker and do not accept others' statements without satisfactory proof."
- 9 -> "You have found it unwise to be too frank in revealing yourself to others."

Write a program that given two different inputs (a data in format DDMMYY, only numbers) and a name (only ASCII characters), computes the previously described algorithm and outputs the string associated to the horoscope.

Example 1

Input

```
090219
Sofia
```

Output

```
At times you have serious doubts as to whether you have made the right
decision or done the right thing.
```

Example 2

Input

```
020207
Amy Jo Lucy Sue
```

Output

```
Disciplined and self-controlled outside, you tend to be worrisome and
insecure inside.
```

Python3

```
frasesProfundas=[
        "You tend to be critical of yourself.",
        "You have a great deal of unused capacity which you have not turned to your
advantage.",
        "While you have some personality weaknesses, you are generally able to compe
nsate for them.",
        "At times you are extroverted, affable, sociable, while at other times you a
re introverted, wary, reserved.",
        "Disciplined and self-
controlled outside, you tend to be worrisome and insecure inside.",
        "At times you have serious doubts as to whether you have made the right deci
sion or done the right thing.",
        "You prefer a certain amount of change and variety and become dissatisfied w
hen hemmed in by restrictions and limitations.",
        "You pride yourself as an independent thinker and do not accept others' stat
ements without satisfactory proof.",
        "You have found it unwise to be too frank in revealing yourself to others."]

def reduct(numberString):
    if len(numberString)==1:
        return int(numberString[0])
    else:
        return reduct(str(sum([int(i) for i in numberString])))

fecha=input()
nombre=input()

finalCode=reduct(str(reduct(fecha))+str(reduct(str(sum([ord(i) for i in nombre])))))

print(frasesProfundas[finalCode-1])
```

16 Skin Cancer

9 points

Introduction

Computer vision is a broad field that will increase in the coming years. Automated disease diagnosis will also be an important field. Today we will merge both, making an algorithm that diagnoses skin cancer. For this purpose, the algorithm has to evaluate the average of the values obtained and their standard deviation (stdev).

Remember that the standard deviation formula is:

$$s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2},$$

where *xi = {x1, x2, x3, ...}* are the observed values of the sample items, $\bar{x}$ is the mean value of these observations, and N is the number of observations in the sample.

Your algorithm will be fed with images from a microscope composed by an array of 5x5 grayscale pixels. (0: means a dark pixel and 255: a white one).

On this simple detector, we can consider a skin cancer if $40 \leq$ average ≤ 80 and stdev ≥ 10. If stdev is lower it's a benign skin mole (benign is a medical term to indicate non-hazardous).

It is also considered a benign skin mole if $80 <$ average ≤ 230. Independent of stdev.

And microscope calibration is needed when: average < 40 or average > 230. Independent of stdev.

HINT: perform all operations as floats.

Input

The input consists of one line of 25 integers separated by spaces.

Output

Print out one of the following outputs classification:

- Skin cancer

- Benign skin mole
- Recalibrate microscope

Example 1

Input

```
50 70 80 40 20 70 50 30 40 60 50 70 80 40 20 70 50 30 40 60 50 70 80 40 20
```

Output

```
Skin cancer
```

Example 2

Input

```
10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
```

Output

```
Recalibrate microscope
```

Example 3

Input

```
90 90 80 90 90 100 100 100 100 110 110 110 110 110 100 100 100 100 100 120
120 120 120 120 120
```

Output

```
Benign skin mole
```

C++

```cpp
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
    int pixels[25];
    int sum=0;
    float average=0, variance=0,stdev=0;

    for (int i=0; i<25; ++i) {
        cin >> pixels[i];
        sum+= pixels[i];
    }

    average=(float)sum/25;

    for(int i = 0; i < 25; ++i){
       variance += (pow((float)pixels[i] - average, 2))/25;
    }

    stdev=sqrt(variance);

    if ( average>=40 && average<=80 && stdev>=10) cout<<"Skin cancer";
    if ( average>=40 && average<=80 && stdev<10) cout<<"Benign skin mole";
    if ( average>80 && average<=230) cout<<"Benign skin mole";
    if ( average<40 || average>230) cout<<"Recalibrate microscope";
}
```

Python3

```python
valuesStr = input().split()
values = [float(i) for i in valuesStr]

avg = sum(values) / len(values)
variance = sum([((x - avg) ** 2) for x in values]) / len(values)
stdev = variance / 2

if avg >= 40 and avg <= 80 and stdev >= 10:
  print('Skin cancer')
elif avg >= 40 and avg <= 230:
  print('Benign skin mole')
else:
  print('Recalibrate microscope')
```

17 Periodic numbers

10 points

Introduction

A periodic number is a number where the same set of digits repeat themselves. That is the case of 1212 or 352352352. A single-digit number is considered as a non-periodic number. Can you write a simple program to find out if a number is periodic or not?

Input

The input will be a single positive integer number.

Output

The output will print a single line stating if the number is periodic:

```
The number x is periodic and its period is y
```

Otherwise, the output will be:

```
The number x is NOT periodic
```

Example 1

Input

```
142857
```

Output

```
The number 142857 is NOT periodic
```

Example 2

Input

```
1212
```

Output

```
The number 1212 is periodic and its period is 12
```

C++

```
#include <iostream>
#include <string>

int main()
{
    std::string number;
    std::cin >> number;
    std::string period;

    bool isPeriod = false;
    for (int i=1; i < number.length(); ++i)
    {

        if (number.length() % i == 0)
        {
            period = number.substr(0, i);
            int j;
            for (j=period.length(); j < number.length(); j = j + period.length())
            {
                if(std::stoi(period) != std::stoi(number.substr(j, period.length())
) )
                {
                    break;
                }
            }
            if (j == number.length())
            {
                isPeriod = true;
                break;
            }
        }

    }
    if (isPeriod)
        std::cout << "The number " << number << " is periodic and its period is " <<
 period << std::endl;
    else
        std::cout << "The number " << number << " is NOT periodic" << std::endl;
}
```

Python3

```
number = str(int(input()))
found = False

# starting from 1 is to avoid the 1char repetition
for ndigit in range(1, len(number)):
    # for every subset of digits count its ocurrences in string
    subset_to_search = number[0:ndigit]
    subsets_found = number.count(subset_to_search)

    # if occurrences times its length is the total number length: found!
    if subsets_found * len(subset_to_search) == len(number):
    # Statement comment 2:
    #   Specify digits (one digit number or a number of one digit repeated)
    #   the following line will discrimine the cases in which digits are the same in
 a subset.
    #         and subset_to_search.count(subset_to_search[0]) != len(subset_to_search
):
        found = True
        print("The number", number, "is periodic and its period is", subset_to_searc
h)
        break

if found != True:
    print("The number", number, "is NOT periodic")
```

18 Fischer Clock

10 points

Introduction

Bobby Fischer is considered one of the greatest chess players of all time. He won the World Chess Championship in 1972 against Boris Spassky. Publicized as a Cold War confrontation between the USA and the USSR, it attracted more worldwide interest than any chess championship before or since.

After that championship Fischer disappeared from the competition and his life became erratic. But in 1988 he filed a patent for a new type of digital chess clock. As you may know, game clocks are used while playing chess tournaments to keep track of the total time taken by the players for their moves. The innovation patented by Fischer was a digital clock that gave each player a fixed time at the start of the game and then added a small amount after each move. For example, if the delay is five seconds and the player has ten minutes remaining on his clock and spent thirty seconds thinking and completing the chess move, he now has 9 minutes and thirty-five seconds remaining. Time can be accumulated, so if the player moves within the delay period, his remaining time increases. This made sure that the players would never be desperately short of time. This invention was officially adopted by the FIDE (World Chess Federation) in 1998.

Given a time control description of the game and a series of game moves, can you write a program to report the actual remaining time for each player?

Input

The input consists of several lines. The first line describes the time control of the game in seconds: the total time per player + delay time. The next lines contain a pair of values specifying the movement time of player 1 and player 2 in seconds with millisecond resolution. Note that the first of this sequence will be the pair 0.0 0.0 as the starting time (delay is not applied). The input ends with the character #.

Output

The remaining time for each player, within 3 decimal precision.

Example

Input

```
900+5
0.0 0.0
7.359 3.237
3.914 2.005
6.077 2.229
4.204 3.128
7.525 5.014
#
```

Output

```
895.921 909.387
```

Python3

```python
line1 = input()
totalTime = float(line1.split("+")[0])
delay = float(line1.split("+")[1])

tPlayer1 = totalTime
tPlayer2 = totalTime

input()
newLine = input()

while newLine!="#":
    tPlayer1 = tPlayer1 + delay - float(newLine.split(" ")[0])
    tPlayer2 = tPlayer2 + delay - float(newLine.split(" ")[1])
    newLine = input()

print("{:.3f}".format(tPlayer1) + " " + "{:.3f}".format(tPlayer2))
```

19 Image Scale

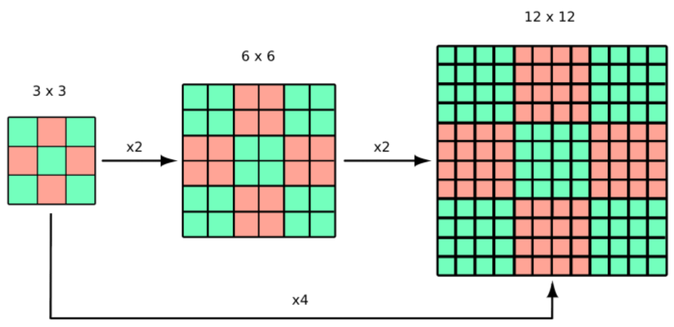
10 points

Introduction

Resizing images is common when dealing with them on computers. You probably have done it hundreds of times. But, how does it work?

Usually, we have the information the file offers and nothing else. So the resolution won't be increased if we make it bigger. Instead, each pixel will be expanded in an area of more pixels. We will be treating the scaling up cases only. Let's get a clearer idea with an example:

Let's say we have a squared image of 9 pixels of 3x3 and want to make it twice as big. Now we will have an image with 36 pixels of 6x6 but with the same information.



See that when we duplicate the size of an image, each pixel now is contained within 4. So we can expect that 1px scaled by a factor of 2 is represented by a 2x2px area.

Now that we already know how images are resized, it is time to actually program it.

Input

- N rows and M columns the image has. Values don't have to be equal.
- Sn, Sm scale up factor for the *n* rows and *y* columns respectively. It must be an integer and values don't have to be equal.
- Each line will be a row of the image. Each pixel will be an integer within the range 0 to 255 separated by a space.

Output

The image value array scaled up.

Example 1

Input

```
3 3
2 2
255 0 255
0 255 0
255 0 255
```

Output

```
255 255 0 0 255 255
255 255 0 0 255 255
0 0 255 255 0 0
0 0 255 255 0 0
255 255 0 0 255 255
255 255 0 0 255 255
```

Example 2

Input

```
2 3
3 2
1 2 3
4 5 6
```

Output

```
1 1 2 2 3 3
1 1 2 2 3 3
1 1 2 2 3 3
4 4 5 5 6 6
4 4 5 5 6 6
4 4 5 5 6 6
```

Python3

```python
N, M = tuple(int(x) for x in input().rstrip().split(' '))
Sn, Sm = tuple(int(x) for x in input().rstrip().split(' '))

image = []
for i in range(N):
    image.append(input().rstrip().split(' '))

for row  in image:
    for j in range(Sn):
        for col in row:
            for k in range(Sm):
                print(col, end=' ')
        print()
```

20 Task Manager

11 points

Introduction

Adrià is in charge of evaluating the problems proposed for this year's edition of CodeWars. Most of these problems are designed by experts inside HP, and when they send them to be supervised, they want to know if the problem submitted is valid or if it needs changes. These experts are very busy and need to get an answer as soon as possible.

Luckily for Adrià, he programmed a few weeks ago an AI that estimates the time it would take him to supervise a problem. However, Adrià is a bit lazy and doesn't want to figure out, every day, the order of the tasks.

Could you give a hand to him? He needs to find a way to sort the problems so that the total time waited by all experts is the minimum.

HINTS:

- Keep in mind Adrià is not a superhero, he can only supervise one problem at a time.
- In case two problems have the same priority, sort in the order they were read from input.

Input

The input will consist of two lines

The first line contains the names of the problems

The second line contains integers with the time needed to evaluate the problem. Each number corresponds to a unique problem.

Realize that both lines must have the same number of elements.

Output

The output should be a list of problems sorted by priority with the time they had to wait and the total time waited.

Example 1

Input

```
Task1 Task2
3 2
```

Output

```
Task2 wait time: 2
Task1 wait time: 5
Total time waited: 7
```

Example 2

Input

```
Prob1 Prob2 Prob3 Prob4
3 1 1 2
```

Output

```
Prob2 wait time: 1
Prob3 wait time: 2
Prob4 wait time: 4
Prob1 wait time: 7
Total time waited: 14
```

Example 3

Input

```
A B C D E F G H I J K L
9 2 5 3 1 1 4 10 3 5 4 3
```

Output

```
E wait time: 1
F wait time: 2
B wait time: 4
D wait time: 7
I wait time: 10
L wait time: 13
G wait time: 17
K wait time: 21
C wait time: 26
```

```
J wait time: 31

A wait time: 40

H wait time: 50

Total time waited: 222
```

C++

```
#include <iostream>
#include <vector>
#include <string>
#include <utility>
#include <algorithm>

int main ()
{
    std::vector< std::pair<std::string, int> > tasks;

    while (std::cin.peek() != '\n')
    {
        std::pair<std::string, int> task;
        std::cin >> task.first;
        tasks.push_back(task);
    }

    for (int i = 0; i < tasks.size(); ++i)
    {
        std::cin >> tasks[i].second;
    }

    std::sort(tasks.begin(), tasks.end(),
        [](std::pair<std::string, int> a, std::pair<std::string, int> b)
        {
            return a.second < b.second;
        });

    int totalTime = 0;
    int currentTime = 0;
    int remainingTasks = tasks.size();

    for (int i = 0; i < tasks.size(); ++i)
    {
        std::cout << tasks[i].first << " wait time: " << currentTime + tasks[i].
second << std::endl;
        currentTime += tasks[i].second;
        totalTime += tasks[i].second * remainingTasks;
        --remainingTasks;
    }
    std::cout << "Total time waited: " << totalTime << std::endl;
}
```

Python3

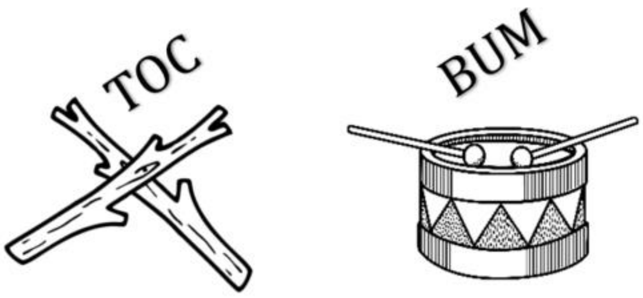
```
keys = input().split(" ")
values = input().split(" ")

tasks = {}

for key in keys:
    tasks[key] = int(values[0])
    values.pop(0)

tasksSorted = {k: v for k, v in sorted(tasks.items(), key=lambda x: x[1])}
time = 0
totalTime = 0

for task in tasksSorted:
    time = time + tasksSorted[task]
    print(task + " wait time: " + str(time))
    totalTime = totalTime + time

print("Total time waited: " + str(totalTime))
```

21 Toc-Bum

11 points

Introduction

The Toc-Bum is a simple mathematical game played during the first years of school. Maybe you remember it from your school years and this problem will provide you with good memories.



The game is usually played by a small group of girls and boys in a circle and it consists of each participant counting a number out loud starting from number 1. Before starting a couple of numbers are defined as special, they are the TOC and the BUM numbers. For example, the 5 is replaced by the TOC and the 7 by the BUM. Therefore, counting numbers aloud must quote each natural number by its name except those that can be composed by a series of TOCs and BUMs.

If we define the number 5 as TOC and the number 7 as BUM we would have this sequence:

1, 2, 3, 4, TOC (5), 6, BUM (7), 8, 9, TOC TOC (since 5 + 5 = 10), 11, TOC BUM (5 + 7 = 12),13, BUM BUM (7 + 7 = 14), TOC TOC TOC (5 + 5 + 5 = 15),16, TOC TOC BUM (5 + 5 + 7 = 17), 18, TOC BUM BUM (5 + 7 + 7 = 19), TOC TOC TOC TOC (5 + 5 + 5 + 5 = 20), BUM BUM BUM (7 + 7 + 7 = 21), ...

Can you write a simple program that given three different positive numbers (the TOC, the BUM and a target number) outputs the corresponding number following the TOC-BUM rule.

HINT: In some cases, there are several solutions for a given target number. Consider this example, 6 as TOC, 8 as BUM and 40 as the target number. The possible solutions are: TOC TOC TOC TOC BUM BUM (6 + 6 + 6 + 6 + 8 + 8) BUM BUM BUM BUM BUM (8 + 8 + 8 + 8 + 8) In such cases the solution considered as valid (in bold) is the one that first tries to use only the TOC number, secondly tries to combine the usage of TOC and BUM numbers and finally tries to use only the BUM number.

Input

The input will contain three lines with three different positive numbers greater or equal than 3. The first number will be the TOC, the second number will be the BUM and the third one the target number.

Output

Print out the TOC-BUM sequence to represent the number or if it is not possible just output the target number.

Example 1

Input

```
5
7
9
```

Output

```
9
```

Example 2

Input

```
5
7
19
```

Output

```
TOC
BUM
BUM
```

Example 3

Input

```
3
4
6
```

Output

```
TOC
TOC
```

Python3

```
toc = int(input())
bum = int(input())
target = int(input())

if (target % toc == 0):
    times = target//toc
    for i in range(times):
        print("TOC")
else:
    gg = target - bum
    n_bum = 1
    if (gg < 0):
        print(target)
    else:
        while (gg % toc != 0):
            gg -= bum
            n_bum += 1
            if (gg < 0):
                print(target)
                break
        if (gg > 0):
            for i in range(gg//toc):
                print("TOC")
        if (gg >= 0):
            for i in range(n_bum):
                print("BUM")
```

22 Automatic Tuner

12 points

Introduction

Did you know that 'Autotune' is one of the most commonly used 'special effects' in the music industry? Among different applications, Autotune is used by many singers and artist to tune their pitch when singing, even in real-time!

As you may know, sounds are acoustic waves that propagate through air. The musical notes we hear are acoustic waves of a certain (and also well known!) vibration frequency. You may know that the seven notes are: Do (C), Re (D), Mi (E), Fa (F), Sol (G), La (A), Si (B). You can check the 'standard' frequency of those notes in the following table:

	C	D	E	F	G	A	B
Reference Frequency (Hz)	261.6	293.7	329.6	349.2	392.0	440.0	493.9

Musical notes have different tones. The same note can sound high-pitched or low-pitched, while still being the same note (those notes are called the octaves). In terms of frequency, it is multiplied or divided by 2 when going to a higher or a lower pitch. See the following table, where different pitches are shown for the note A, taking 440 Hz as the reference pitch:

Octave	Factor	A
2nd High Octave	4	1760.0
1st High Octave	2	880.0
Reference Frequency (Hz)		440.0
1st Low Octave	1/2	220.0
2nd Low Octave	1/4	110.0

Note that this table shows only some of the pitches for the note A, but there are many more.

But, when we sing or play an instrument, the note we sing or play may not be perfectly pitched. That means that if you try to sing a 'La', you may be generating a sound close to the 'La' frequency, but not exactly that note. Autotune corrects it to make it sound like a perfectly pitched note!

So, in this problem, you are asked to implement a simple autotune program. Given an input frequency, output the identified note with the proper octave and also the correction needed, in Hz, to make it sound perfectly pitched.

Input

One line of a decimal positive value bounded between 20.0 and 20,000.0.

Output

The output of this problem is made by three lines:

- First line the identified note corresponding to the input frequency
- Second line the closest note with the proper octave frequency
- Third line the correction to make the input frequency sound perfectly pitched. If the input frequency is already perfectly pitched, then the output may be slightly different. Check the examples!

In case the frequency given lays just in the middle of two notes, it is preferable to increase the frequency instead of decreasing.

All values must be rounded to one decimal.

Example 1

Input

```
440.0
```

Output

```
Input note: A (440.0 Hz)
Closest note: A (440.0 Hz)
Pitch Perfect!
```

Example 2

Input

```
1245.2
```

Output

```
Input note: D (1245.2 Hz)
Closest note: D (1174.8 Hz)
Decrease frequency in 70.4 Hz
```

Example 3

Input

```
15062.2
```

Output

```
Input note: B (15062.2 Hz)
Closest note: B (15804.8 Hz)
Increase frequency in 742.6 Hz
```

Python3

```python
import math
freqs = [261.6, 293.7, 329.6, 349.2, 392.0, 440.0, 493.9]
notes = ['C', 'D', 'E', 'F', 'G', 'A', 'B']

f = float(input())
octave = [math.log(f/freq, 2) for freq in freqs]
error = [abs(f - freqs[i]*(2**round(octave[i]))) for i in range(len(octave))]
noteI = error.index(min(error))

print(f"Input note: {notes[noteI]} ({f:.1f} Hz)")
ff = freqs[noteI]*(2**round(octave[noteI]))
print(f"Closest note: {notes[noteI]} ({ff:.1f} Hz)")
diff = ff - f
if diff == 0:
    print("Pitch Perfect!")
elif diff < 0:
    print(f"Decrease frequency in {abs(diff):.1f} Hz")
else:
    print(f"Increase frequency in {abs(diff):.1f} Hz")
```

23 Mario Prize

12 points

Introduction

Mario has just won a contest on a website where he can choose two prizes from a given list. Although he won't have to pay for the prizes, if the total weight exceeds 3kg, he will have to pay shipping taxes.

Mario doesn't have any money, so he will have to choose a pair of prizes that together weigh 3kg or less. The website also allows the winner to exchange the chosen products for money. Let's help him maximize the benefits by creating a program that, given the following data table, returns which two objects from the list have the highest combined worth, as long as the combined weight does not exceed 3kg.

Products offered (example):

Product	Price (€)	Weight (Kg)
Laptop	2300	2,5
TV 64"	4500	13,8
Multifunction printer	250	6,4
Pocket printer	160	0,3
Backpack	30	0,9
Headset wireless	250	1,3
Pendrive 128 Gb	140	0,2
Keyboard	80	1,2
Mouse	150	0,5
Projector	2400	4,6

Input

- Vector containing 10 price values
- Vector containing 10 weight values

Output

Mario chooses: prize A and prize B with a total profit of value EUR

NOTE: there will only be one solution. Print prize A and prize B sorted by the index value (A < B).

Example

Input

```
2300 4500 250 160 30 250 140 80 150 2400
2.5 13.8 6.4 0.3 0.9 1.3 0.2 1.2 0.5 4.6
```

Output

```
Mario chooses: prize 0 and prize 3 with a total profit of 2460 EUR
```

Python3

```python
values = input().split(" ")
weights = input().split(" ")
values = [float(i) for i in values]
weights =[float(i) for i in weights]

maxValue = 0
maxI = 0
maxJ = 0

for i in range(0,len(weights)):
    for j in range(i+1,len(weights)):
        if weights[i] + weights[j] <= 3:
            if values[i] + values[j] > maxValue:
                maxValue = values[i] + values[j]
                maxI, maxJ = i, j

print("Mario chooses: prize " + str(maxI) + " and prize " + str(maxJ) + " with a
 total profit of " + str(int(maxValue)) + " EUR")
```

24 Flower classification

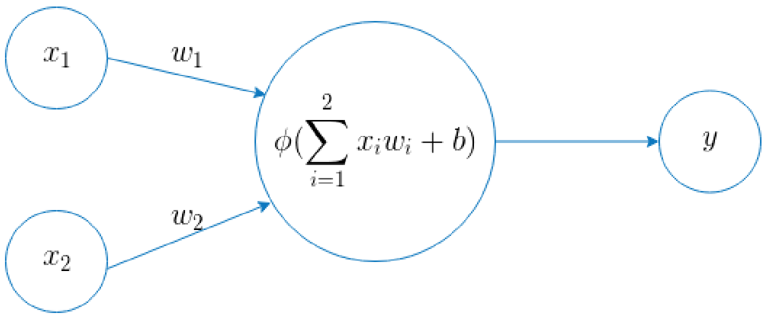
12 points

Introduction

Your parents have a beautiful garden full of flowers. Although there are only two types of flowers, red and blue, their color isn't strong enough to tell them apart. In order to classify them, you are given a spreadsheet containing information about the length and width of the petals of each flower. The spreadsheet is as follows:

length	width	type
3	1.5	red
2	1	blue
4	1.5	red
3	1	blue
3.5	0.5	red
5.5	1	?
1	1	?
2	1.4	?

Here, your parents have already filled in some values. Your job is to fill in the rest of the table. To speed up this process we can use machine learning algorithms such as neural networks!



This one is called a perceptron. It is the simplest type of neural network, but for our task it is enough. It can produce good results.

This neural network, has two inputs: *x1* and *x2*. In our case, they correspond to the length and width of the flower. The network performs a mathematical operation on those input values and gives you the answer *y*. In this case *y* can be "blue" or "red".

In order to implement this algorithm in code, you must follow these steps:

1. The formula inside the middle neuron can be written as:

$$\sum_{i=1}^{2} x_i w_i + b = x_1 w_1 + x_2 w_2 + b$$

Where *x1* is the length and *x2* is the width.

w1, *w2* and *b* are some magic values that the network learns by itself when it is trained and is told how to solve a given problem.

2. Next, once you have multiplied and added the values in the above formula, the result should be plugged into the next formula:

$$\phi(x) = \frac{1}{1 + e^{-x}}$$

3. Finally, if the result is smaller than 0.5, the flower is blue; otherwise the flower is red.

Getting the values of *w1*, *w2* and *b* is not an easy task. To make it simple someone already calculated those values for you:

```
w1 = 1.9116881315655996
w2 = 1.22872621999026
b = -7.119940111025795
```

If you don't remember Euler's number (*e*) by heart, here it is:

```
e = 2.718281828459045
```

Input

The input consists of:

- The first line is a positive integer, *N*, indicating the number of flowers.
- The following lines have the length and width of the *N* flowers, with this format: length width (separated by a space).

Output

For each flower length and width input, the output must be:

- If the flower is blue:

```
The flower (<length> <width>) is blue
```

- If the flower is red:

```
The flower (<length> <width>) is red
```

Example 1

Input

```
1
3 1.5
```

Output

```
The flower (3 1.5) is red
```

Example 2

Input

```
3
2 1
5.5 1
3.5 0.5
```

Output

```
The flower (2 1) is blue
The flower (5.5 1) is red
The flower (3.5 0.5) is red
```

C++

```
#include <iostream>
#include <cmath>

int main(void)
{
    int n;

    std::cin >> n;

    for(int i = 0; i < n; ++i)
    {
        double l, w;

        const double w1 =  1.9116881315655996;
        const double w2 =  1.22872621999026;
        const double b  = -7.119940111025795;

        std::cin >> l >> w;

        double y = l * w1 + w * w2 + b;
        double sigmoid = 1 / ( 1 + exp(-y) );

        std::cout << "The flower (" << l << ' ' << w << ") is " <<  ( sigmoid<0.
5 ? "blue" : "red" ) << std::endl;
    }
}
```

25 Back To The Future

13 points

Introduction

Hello time traveller!

You are in big trouble: the machine you've just travelled with is broken. Time is going back, and everything is going to disappear. There is only one way to fix it.

You had an analogical clock that your grandpa gifted you, and the clock hands are going backwards as well. To repair the space-time fissure, you have to determine the angle between the clock hands when the light flashes in the machine.

Hurry up! There is no time... or is there?

Input

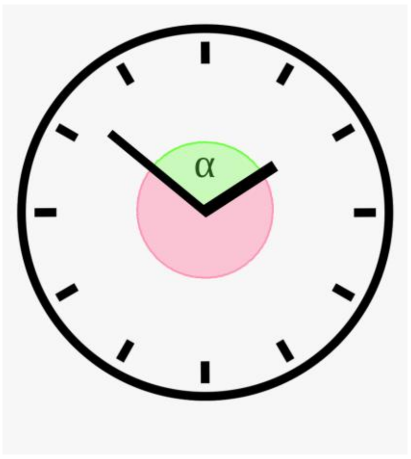
The times when the machine flashes a light are given in each line in the HH:MM format; hours only go from 1 to 12 because this is an analog clock.

Also remember, because this is an analog clock, the hour hand moves between the hour marks when the time is not a full hour (between 4:00 and 5:00 o'clock, for example).

A hash sign '#' indicates End Of File.

Output

Each line is the acute (smaller) angle between the hands (colored green in the image below), as an absolute value, rounded to one decimal place. It does not matter which hand is on the right or left.



Example

Input

```
07:23
02:22
04:00
06:15
#
```

Output

```
83.5
61.0
120.0
97.5
```

C++

```cpp
#include <iostream>
#include <iomanip>
#include <cmath>

int main()
{
    std::cout << std::fixed << std::setprecision(1);
    std::tm t = {};
    while (std::cin >> std::get_time(&t, "%I:%M"))
    {
        double h = (t.tm_hour)%12 + t.tm_min / 60.0;
        double ah = h * 360.0 / 12.0;
        double am = t.tm_min * 360.0 / 60.0;
        double a = std::abs(ah - am);

        if( a>180.0 )
        {
            a -= 360.0;
            a = -a;
        }

        std::cout << std::round(a*10.0)/10.0 << std::endl;
    }
}
```

26 Autocomplete

13 points

Introduction

As you may know, most text messaging apps use autocomplete mechanisms in order to predict what you are trying to type. Commonly the autocomplete is also customized according to the user's usual speech patterns.

We are developing what we expect to be the next most popular text messaging app, and this feature is a must!

Those mechanisms can be based in very complex algorithms, but since this is our first version, let's make it simpler:

The user will have typed some text to which we have access. He or she has also started typing the next word, the one we must predict.

Taking into account the contents of the given text, our app must suggest the next word to be typed.

To clarify how this should work, our manager has given us the following rules:

- The suggested word must start with the same characters the user has typed.
- If there is no word that satisfies the previous rule, we suggest the same 'word' the user has typed.
- If we have more than one word starting with the user's input, we suggest the most common one.
- In case two or more suggestions are equally common, we must suggest the one that comes first in alphabetical order.

Input

- Text in one line.

> HINT: be careful with capital letters ('Printer' must be treated the same as 'printer'). Also, we can expect to only have alphabetical characters.

- The word we want to autocomplete. To make things a little bit easier, the word we want to autocomplete will only contain lowercase letters.

Output

Suggestion for the input word, in lowercase.

Example 1

Input

```
How many cookies could a good cook cook If a good cook could cook cookies
coo
```

Output

```
cook
```

Example 2

Input

```
this is just a normal text
hou
```

Output

```
hou
```

Example 3

Input

```
animal animal animal animate animate animate
anima
```

Output

```
animal
```

Python3

```python
sentence = input().lower().split(" ")
inputWord = input().lower()

suggestions = {}

for word in sentence:
    if word[0:len(inputWord)] == inputWord:
        if word in suggestions:
            suggestions[word] = suggestions[word] + 1
        else:
            suggestions[word] = 1

suggestions = dict(sorted(suggestions.items(), key=lambda kv: kv[0]))
if suggestions!= {}:
    print(max(suggestions, key=suggestions.get))
else:
    print(inputWord)
```

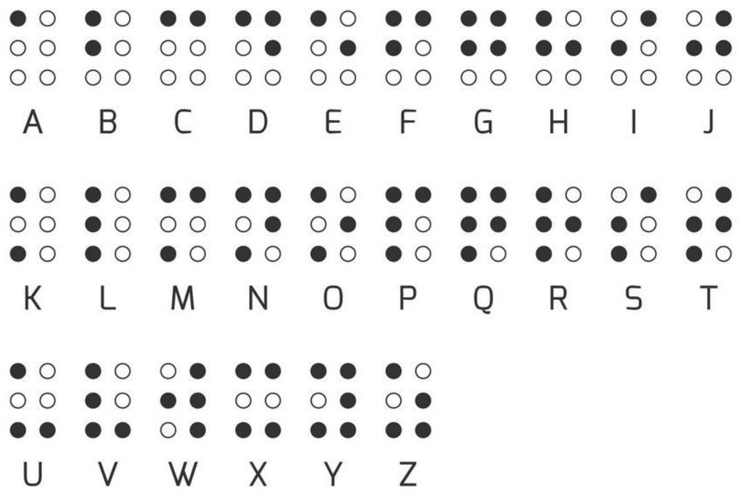
27 Tactile Writing

15 points

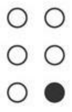
Introduction

Louis Braille developed in 1829 a tactile writing system for visually impaired people. This system represents symbols with units of space known as braille cells. A full braille cell consists of six raised dots arranged in two columns, with three dots in each. A total of sixty-four combinations are possible using one or more of these six dots. A braille cell can be used to represent an alphabet letter, number, punctuation mark, or even a whole word.

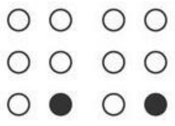
These are the braille symbols for the English braille alphabet:



Although braille does not have a separate alphabet of capital letters as there is in print writing systems, a capital letter is designated by placing the following cell in front of a letter:



To capitalize a whole word, place two of these in front of the word:



Can you code a simple program to translate a text written in regular English characters to the Braille English alphabet? The filled and empty dots will be replaced by the characters "*" and "." respectively. And do not forget about processing any white space in the input text.

Input

A message written in English characters.

Output

The message translated to the Braille English alphabet.

Example 1

Input

```
HELLO WORLD
```

Output

```
....*.*.*.*.*. .....**.*.*.**
....**.**.*..* ....**.****..*
.*.*....*.*.*. .*.*.**.*.*...
```

Example 2

Input

```
This is sample written in Braille
```

Output

```
...**..*.* .*.* *. .**.*****.*. .**..*.*.**.** .*** ..*.*.*..**.*.*.
..*****.*. *.*. .. *.....*.*..* *****.****.*.* *..* ..*.**..*.*.*..*
.**.....*. ..*. .. *...*.*.*... .**...*.*...*. ..*. .*..*.....*.*...
```

Python3

```
# Auxiliar function to check is a word is capitalized, that is only the first ch
aracter is uppercase

def isCapitalized(s):
  """iscapitalized(S) -> bool

  Returns True if S is a capitalized string and there is
  at least one character in S.
  """
  return s and (s.capitalize() == s)

# This is the braille alphabet

braille_alphabet = { "A":"*.....",
                     "B":"*.*...",
                     "C":"**....",
                     "D":"**.*..",
                     "E":"*..*..",
                     "F":"***...",
                     "G":"****..",
                     "H":"*.**..",
                     "I":".**...",
                     "J":".***..",
                     "K":"*...*.",
                     "L":"*.*.*.",
                     "M":"**..*.",
                     "N":"**.**.",
                     "O":"*..**.",
                     "P":"***.*.",
                     "Q":"*****.",
                     "R":"*.***.",
                     "S":".**.*.",
                     "T":".****.",
                     "U":"*...**",
                     "V":"*.*.**",
                     "W":".***.*",
                     "X":"**..**",
                     "Y":"**.***",
                     "Z":"*..***" }

# Braille does not have a separate alphabet of capital letters as there is in pr
int.
# Capital letters are indicated by placing a dot 6 (.....*) in front of the lett
er to be capitalized.
# Two capital signs mean the whole word is capitalized.
# Read the original message from input
original_message = input()

# Prepare structure to store the capitalization level for each word
capitalization_level_words = ""

# Get all the words from the original message
words = original_message.split(" ")

# Check whether the word is uppercase(2), capitalized(1) or lowercase(0)
for i in words:
  if i == "":
    # Discard any empty word at the very beginning of the original message
    continue
  elif i.isupper():
    capitalization_level_words += "2"
  elif isCapitalized(i):
    capitalization_level_words += "1"
  else:
    capitalization_level_words += "0"

# Define the braille output structure
braille_output = ["","",""]

# Now proceed with the conversion of the text checking for each word the capital
ization level to apply.
new_word = True

for i in original_message:
  # Beware of white spaces and remember that after a space a new word could appe
ar
  if i == " ":
    braille_output[0] += "  "
    braille_output[1] += "  "
    braille_output[2] += "  "
    new_word = True
  else:
    # If a new word is read check the capitalization level preprocessed for this
 word
    if new_word:
      cap = capitalization_level_words[0]
      capitalization_level_words = capitalization_level_words[1:]
      # Add the prefix to capitalize the word
      if cap == "1":
        braille_output[0] += ".."
        braille_output[1] += ".."
        braille_output[2] += ".*"
      # Add the prefix to uppercase the word
      elif cap == "2":
        braille_output[0] += "...."
        braille_output[1] += "...."
        braille_output[2] += ".*.*"

    braille_output[0] += braille_alphabet[i.upper()][0:2]
    braille_output[1] += braille_alphabet[i.upper()][2:4]
    braille_output[2] += braille_alphabet[i.upper()][4:6]
    new_word = False

# Finally print the output
print(braille_output[0])
print(braille_output[1])
print(braille_output[2])
```

28 Dilahk's contest

17 points

Introduction

Our colleague Dilahk needs help with his math contest. Here is what the contest is about: given a number n, the students have to draw a triangle of numbers with n levels.

To start building the triangle, start with 1 at the top, then continue placing numbers below it in a triangular pattern. The next row of the triangle is constructed by summing adjacent elements in the previous row. Because there is nothing next to the 1 in the top row, the adjacent elements are considered to be 0.

So, the simplest Dilahk's triangle has only 1 level:

```
1
```

And here is the Dilahk's triangle of 2 levels:

```
 1
1 1
```

This process will be repeated to produce each subsequent row. If the given number is 3, then the triangle will look like this one:

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

But Dilahak's triangle is formed exclusively by single-digits. So, when the result of adding adjacent elements in the previous row is greater than 9, all the digits of the result are added until the result is a single-digit number. For instance, if the given number is 6, then the triangle will look like this one:

```
     1
    1 1
   1 2 1
  1 3 3 1
 1 4 6 4 1
1 5 1 1 5 1
```

HINT: Notice that between, after and before the numbers there are white spaces. For instance, in the above example should be written as this: (where - denotes a white space for visual representation)

```
-----1-----
----1-1----
---1-2-1---
--1-3-3-1--
-1-4-6-4-1-
1-5-1-1-5-1
```

Input

A positive integer N (greater than or equal to 1).

Output

You should write the numbers until Nth row in form of a triangle where each position is calculated as explained above.

Example 1

Input

```
1
```

Output

```
1
```

Example 2

Input

```
3
```

Output

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

Example 3

Input

```
6
```

Output

```
     1
    1  1
   1  2  1
  1  3  3  1
 1  4  6  4  1
1  5  1  1  5  1
```

Python3

```
def convert(num):
  res = 0
  if int(num) >= 10:
    digits = len(num)
    for i in range(digits):
      res = res + int(num[i])
  else:
    res = num
  return res

num_levels = int(input())

rows = num_levels
columns = (2*rows)-1

Matrix = [[0 for x in range(columns)] for y in range(rows)]

for i in range(rows):
# Level 0
  if (i==0):
    Matrix[i][columns//2] = "1"
# Level n
  else:
    for j in range(columns):
      if Matrix[i-1][j]!=0:
        Matrix[i][j-1]=str(int(Matrix[i-1][(j-2)%columns]) + int(Matrix[i-
1][j]))
        Matrix[i][j-1]=convert(Matrix[i][j-1])
        Matrix[i][j+1]=str(int(Matrix[i-1][j]) + int(Matrix[i-
1][(j+2)%columns]))
        Matrix[i][j+1]=convert(Matrix[i][j+1])

for i in range(rows):
  for j in range(columns):
     if Matrix[i][j] == 0:
       Matrix[i][j] = " "
     print(Matrix[i][j], end="")
  print()
```

29 Adaptive Mean Thresholding

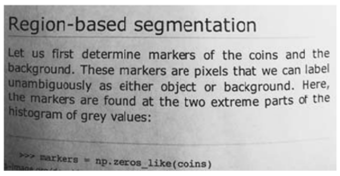
21 points

Introduction

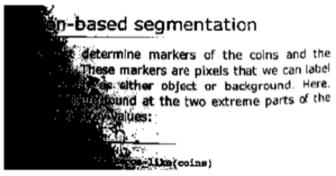
The simplest thresholding methods replace each pixel in an image with a black pixel (value = 0) if the image intensity is less than some threshold value, or a white pixel (value = 1) if the image intensity is greater than that threshold value. This process is also called binarization of an image.

However, applying the same threshold value for the whole image is a very crude method and only works well when the image has no shadows or bright regions. The following is an example of the result of applying a basic thresholding algorithm vs the original image.

Original Image

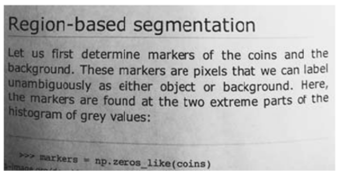


Global Threshold Image

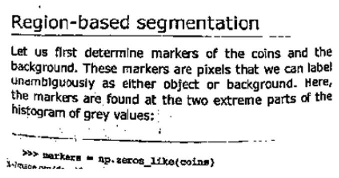


It is possible to improve the performance of thresholding by using Adaptive Thresholding methods. See the following example and note the difference between global and adaptive thresholding.

Original Image



Adaptative Threshold Image



The objective of this problem is to implement an Adaptive Mean Thresholding algorithm. Don't panic! The principle of this algorithm is very easy to understand! Adaptive Mean Thresholding computes a threshold value per each pixel of the image instead of computing a single threshold value for all the pixels.

This individual thresholding value is obtained by calculating the mean of a NxN matrix (also called the 'window') whose center is a certain pixel. In the following example, we have centered the 3x3 window on the pixel colored light green. The pixels inside the 'window' are colored blue.

Image Size = 7 x 7

125	125	75	125	255	75	45
100	100	75	50	40	25	0
100	150	175	190	225	245	255
255	255	250	245	255	255	255
250	245	240	235	225	220	215
200	150	200	100	150	100	50
50	25	20	0	25	25	0

Window Size = 3 x 3

If the pixel value is greater than or equal to the obtained threshold, that pixel is assigned the value 1. If not, that pixel is assigned the value 0. See the following example:

225	220	215
150	100	50
25	25	0

$$1)\ mean\ threshold = \frac{225+220+215+150+100+50+25+25+0}{9} = 112.2 = \mathbf{112}$$

$$2)\ pixel\ value = 100$$

$3)\ pixel\ value \geq mean\ threshold?$ → NO

$$4)\ pixel\ value = 0$$

This process is carried out for each pixel in the image. Corner pixels and pixels situated along the sides of the image are missing some neighboring pixels. When placing the window around these pixels, assume that the value of missing neighbors is '0'. For example:

0	0	0					
0	125	125	75	125	255	75	45
0	100	100	75	50	40	25	0
	100	150	175	190	225	245	255
	255	255	250	245	255	255	255
	250	245	240	235	225	220	215
	200	150	200	100	150	100	50
	50	25	20	0	25	25	0

Write a program that applies the Mean Adaptive Threshold method to an input image given the size of the window.

HINT: Round the computed mean threshold value to the nearest integer before comparing its value with the pixel value!

Input

The number of lines of the input depends on the size of the image, but it always follows the same structure:

- First line is the number of rows of the input image.
- Second line is the number of columns of the input image.
- The following lines correspond to the input image.
- Finally, the last line is the size of the window. This must be an odd value.

Output

The output is the thresholded image.

Example

Input

```
5

5

7 5 8 4 7

3 6 5 5 1

9 3 8 8 9

2 4 9 1 1

4 4 1 4 9

5
```

Output

```
1 1 1 1 1

1 1 1 1 0

1 0 1 1 1

1 1 1 0 0

1 1 0 1 1
```

Python3

```python
# ADAPTATIVE THRESHOLDING PROBLEM - CODEWARS 2020 #

# GET INPUT #
Nrows = int(input())
Ncols = int(input())

img = list()
for k in range(Nrows):
     img.append(input().split())

th_window = int(input())

# PAD IMG WITH ZEROS
pad_width = th_window // 2
pad_rows = Nrows + 2*pad_width
pad_cols = Ncols + 2*pad_width

img_pad = [[0 for i in range(pad_cols)] for j in range(pad_rows)]

for row in range(pad_width, pad_rows - pad_width):
    for col in range(pad_width, pad_cols - pad_width):
        img_pad[row][col] = int(img[row - pad_width][col - pad_width])
```

```
# COMPUTE ADPT THRESH. + BINARIZE #
for row in range(pad_width, pad_rows - pad_width):
    for col in range(pad_width, pad_cols - pad_width):
        # Window coordinates
        row0 = row - pad_width
        col0 = col - pad_width
        rowf = row + pad_width
        colf = col + pad_width

        # Compute average value of pixels in window
        cum_sum = 0
        for n in range(row0,rowf+1):
            for m in range(col0,colf+1):
                cum_sum = cum_sum + img_pad[n][m]

        avg_threshold = round(cum_sum / (th_window**2))

        # Check if pixel is above or below threshold
        row_img = row - pad_width # row index in original input image
        col_img = col - pad_width # col index in original input image
        if int(img[row_img][col_img]) >= avg_threshold:
            img[row_img][col_img] = '1'
        elif int(img[row_img][col_img]) < avg_threshold:
            img[row_img][col_img] = '0'


# PRINT IMG BINARIZED
for row in range(Nrows):
    for col in range(Ncols):
        val = int(img[row][col])
        print(val, end=' ')
    print('\n',end='')
```

30 Viking Assault

24 points

Introduction

You are aboard a warship at sea on a very foggy day, when the enemy ambushes you. The general gives the order to fire, but because of the fog, you can't tell whether you've hit an enemy. However, you know where the enemies are thanks to the boat watch, and you know the angle and distance in which the cannons are fired. Help your general figure out how many enemies you've hit, and how many have been sunk.

Each shot has a blast radius of 5m, and the damage each shot inflicts goes from 100% at the center to 0% at the edge (50% at 2.5m, etc), rounded. If a ship is hit by more than one shot, the damage is cumulative. A ship sinks when it sustains 100% of damage.

Input

On the first line, we are given the number of targets and the number of shots taken, separated by a space.

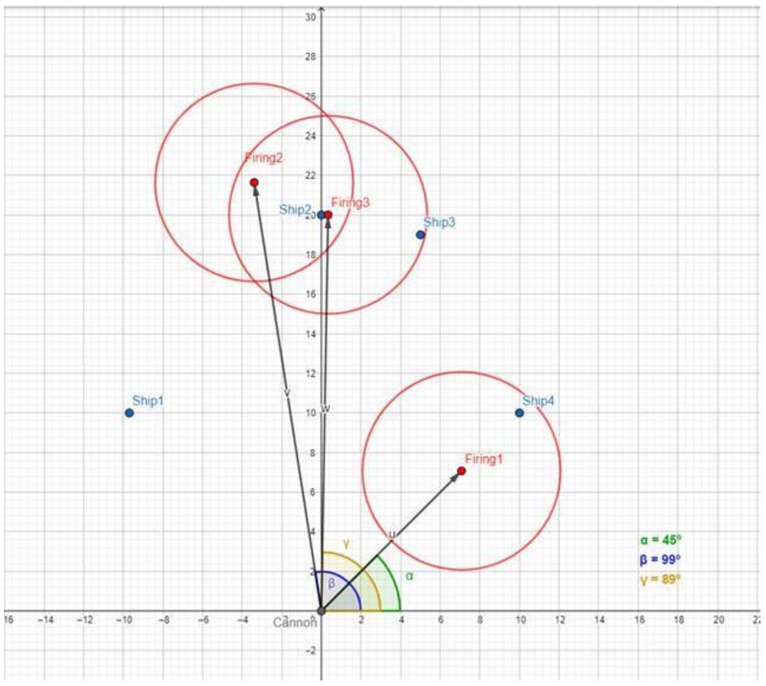
Then, for each target, we are given the coordinates X and Y respectively in each line, separated by a space.

Finally, for each shot, we are given the angle at which the cannon was placed, and the reach of the shot from the ship, separated by a space.

Output

Each output line corresponds to an input target, in the same order the targets were given. If the target has been sunk, the output should be "DESTROYED"; if not, then the output should be the ship's remaining health percentage (as an integer value).

Example



Input

```
4 3
-10 10
0 20
5 19
10 10
45 10
99 22
89 20
```

Output

```
100
DESTROYED
95
83
```

C++

```
#include <iostream>
#include <string>
#include <vector>
#include <cmath>
#include <utility>

#define PI 3.14159265

using namespace std;

struct ship_position_t
{
    int pos_x;
    int pos_y;
};

struct input_info_t
{
    int targets;
    int shots;
    std::vector<std::pair <int,int>>* targets_vector;
    std::vector<std::pair <int,int>>* shots_vector;
    std::vector<int>* ships_health;
};




float calculate_distance(std::pair <int,int> target_pos, std::pair <int,int> sho
t_pos)
{

    double x_shot = shot_pos.first*cos(shot_pos.second*PI/180.0);
    double y_shot =  shot_pos.first*sin(shot_pos.second*PI/180.0);

    double first_term = std::abs(target_pos.first- x_shot);
    double second_term = std::abs(target_pos.second - y_shot);
    first_term *=first_term;
    second_term *=second_term;
    return sqrt(first_term+second_term);
}



void update_ships_status(input_info_t* info)
```

```
{
    //cout << "Insert the number of targets and shots separated by a space\n";
    cin >> info->targets;
    cin >> info->shots;
    //cout << "Insert the coordinates of each target, separating x and y coordin
ates with a comma: x,y\n";
    info->targets_vector = new std::vector<std::pair<int,int>> ();
    info->shots_vector = new std::vector<std::pair<int,int>> ();
    for(int i = 0;i <info->targets;i++)
    {
        int x;
        int y;
        cin >> x;
        cin >> y;
        std::pair <int,int> pos;
        pos = make_pair(x,y);
        info->targets_vector->push_back(pos);
    }
    //cout << "Insert the angle and the distance reached by each shot, separatin
g distance and angle coordinates with a space: a d\n";
    for(int i = 0;i <info->shots;i++)
    {
        int d;
        int a;
        cin >> a;
        cin >> d;
        std::pair <int,int> pos;
        pos = make_pair(d,a);

        info->shots_vector->push_back(pos);
    }

}

void calculate_health(input_info_t* info)
{
    info->ships_health = new std::vector<int> ();
    for(size_t i = 0; i < info->targets_vector->size();i++)
    {
        info->ships_health-
>push_back(100);//initial value of health for ship "i";
        for(size_t j = 0; j < info->shots_vector->size();j++)
        {
            float distance_ij = calculate_distance((*info-
>targets_vector)[i],(*info->shots_vector)[j]);
            if(distance_ij < 5)
            {
                float harm = (5 - distance_ij) / 5 * 100;
                if(harm-static_cast<int>(harm) > 0.5)
                {
                    (*info->ships_health)[i] -= static_cast<int>(harm)+1;
                }
                else
                {
                    (*info->ships_health)[i] -= static_cast<int>(harm);
                }
            }
            if((*info->ships_health)[i] < 0)
            {
                (*info->ships_health)[i] = 0;
            }

        }
    }
}

void print_status(input_info_t* current_info)
{
    vector<int>::iterator i;
    for (i = current_info->ships_health->begin();i < current_info->ships_health-
>end();i++)
    {
        if(*i == 0)
        {
            cout << "DESTROYED\n";
        }
        else
        {
            cout << *i<<endl;
        }
    }
}


int main()
{
    input_info_t* current_info = new input_info_t();
    update_ships_status(current_info);

    calculate_health(current_info);
    print_status(current_info);

    return 0;
}
```

31 Pharaoh Puzzle

25 points

Introduction

As most of you know, the Egyptian pyramids are one of the defining architectural achievements of the ancient world. Ancient Egyptians believed that when the king died, part of his spirit (known as "ka") remained with his body. To properly care for his spirit, the corpse was mummified, and everything the king would need in the afterlife was buried with him, including gold vessels, food, furniture and other offerings.

Last month a new pyramid was discovered, and we are part of the team involved in the first exploration of the building. We found a very strange puzzle, and after solving the missing symbols, we had access to the first room. Every room is protected with one of these puzzles and each puzzle is more difficult to solve than the previous one.

Your main objective is to fill in the missing symbols!

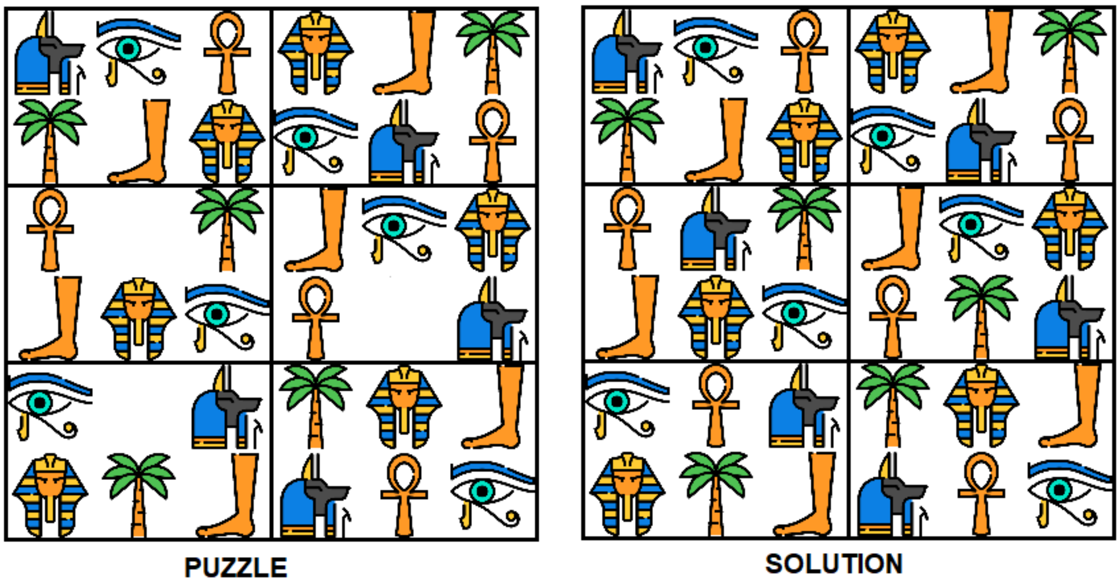


Fig 1. The first puzzle we found and its valid solution.

From solving the first puzzle we discovered that:

- A symbol can appear only once in each row.

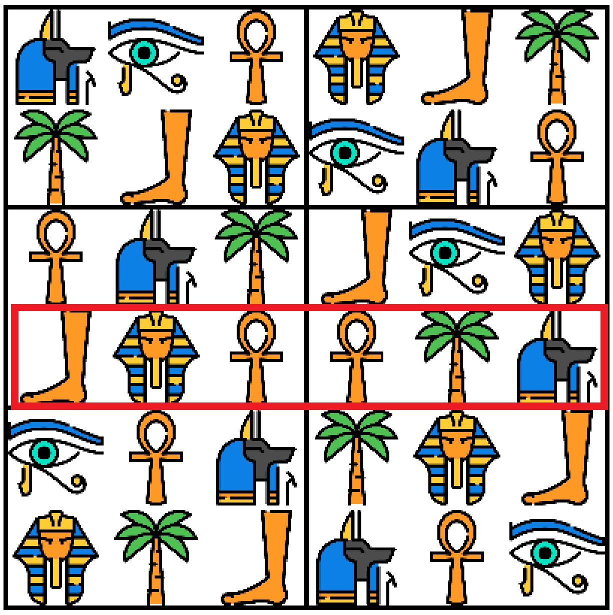


Fig 2. Incorrect solution: the fourth row contains identical symbols.

- A symbol can appear only once in each column.

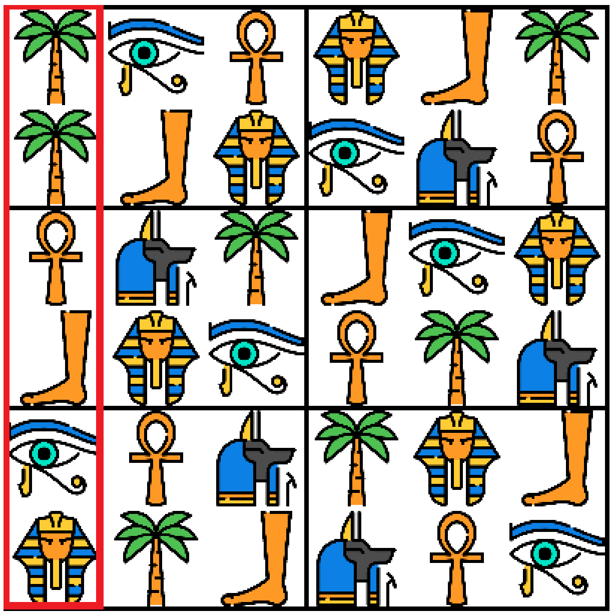


Fig 3. Incorrect solution: the first column contains two identical symbols.

- A symbol can appear only once in each diagonal.

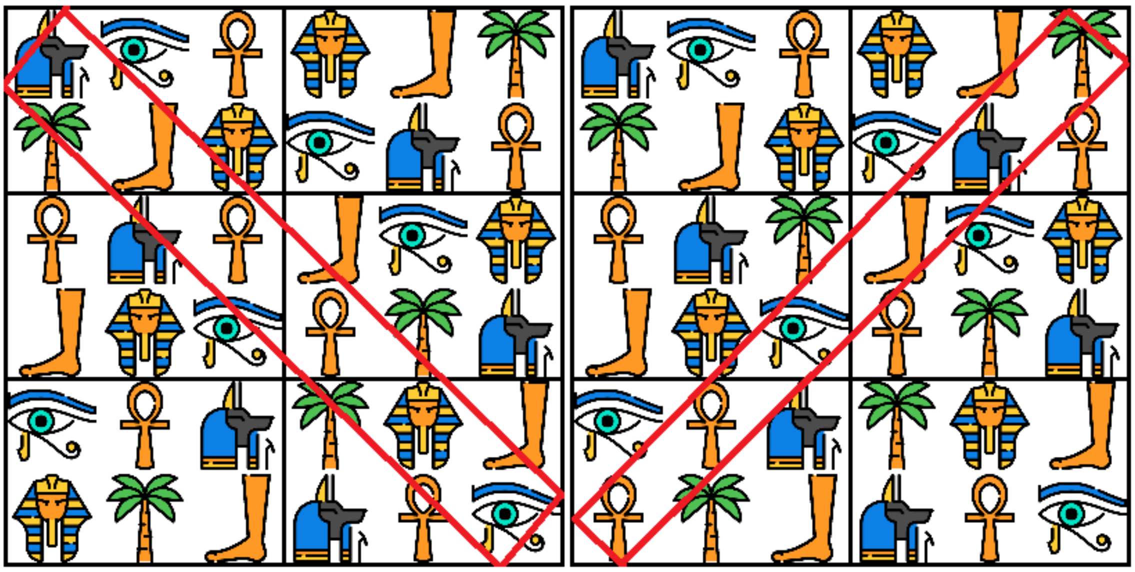


Fig 4. Incorrect solution: both diagonals contain two identical symbols.

- A symbol can appear only once in each 2x3 square.

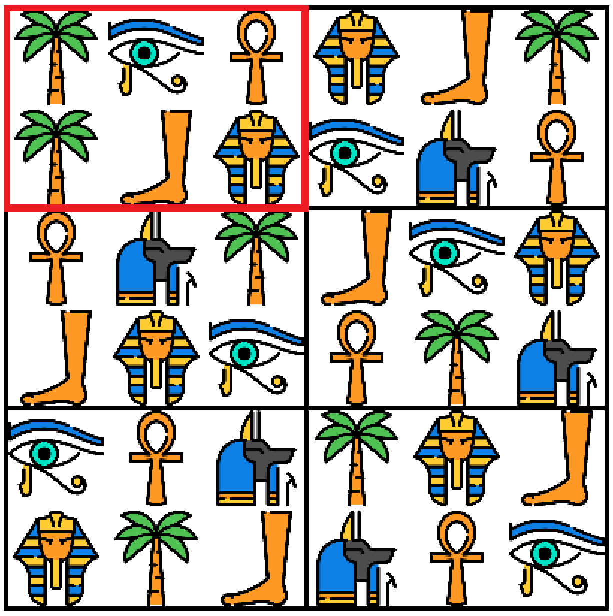


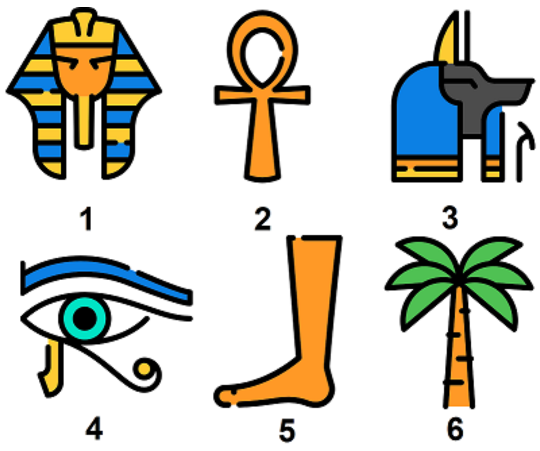
Fig 5. Incorrect solution: the top left square contains two identical symbols.

Please study the examples to avoid confusion. Each input has only one valid solution!

Input

The input of the program is one line with 36 integers, between 1-6 for the symbols, 0 for an empty space, and separated with white spaces. Each set of 6 values represents a table row.

Here is a table for converting the symbols to numbers:



Output

The output is 36 integer between 1-6, separated by white spaces.

Example 1

Input

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

Output

```
6 3 1 2 4 5 5 4 2 1 3 6 2 5 3 4 6 1 4 1 6 5 2 3 3 2 5 6 1 4 1 6 4 3 5 2
```

Example 2

Input

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

Output

```
6 4 1 3 5 2 2 5 3 1 4 6 3 2 4 5 6 1 5 1 6 2 3 4 4 3 2 6 1 5 1 6 5 4 2 3
```

Python3

```
def check(grid_values):
  """
  Check that a list of elements contains exactly one value from 1-6.

  Parameters
  ==========
  grid_values: lst of int
    List of integers representing the symbols grid.
  """
  for i in range(1,7):
    if grid_values.count(i) > 1:
      return False

  return True

def validateGrid(grid_values):
  """
  Validate a grid. Cheking than there are not duplicates elements in any row,
  column, diagonal, and subgrid(2x3 section).

  Parameters
  ==========
  grid_values: lst of int
    List of integers representing the symbols grid.
  """

  #Check row
  for row in range(0,36,6):
      rowCheck = grid_values[row:row + 6]
      if not check(rowCheck):
        return False

  #Check cols
  for col in range(6):
      colCheck = [grid_values[row + col] for row in range(0, 36, 6)]
      if not check(colCheck):
        return False

  # Check sub-grids
  for subgridRow in range(0,6,2):
    for subgridCol in range(0,6,3):
      square = []
      square = square + grid_values[subgridRow * 6 + subgridCol :
                                    subgridRow * 6 + subgridCol + 3]
      square = square + grid_values[(subgridRow + 1) * 6 + subgridCol:
```

```
                                        (subgridRow + 1) * 6 + subgridCol + 3]
        if not check(square):
          return False


    # Check diagonals
    diag1 = []
    diag2 = []

    for idx in range(6):
        diag1.append(grid_values[idx * 6 + idx])
        diag2.append(grid_values[idx * 6 + 5 - idx])

    if (not check(diag1)) and (not check(diag2)):
      return False

    return True

def solveGrid(grid_values):
    """
    If a symbol is missing try to fill it with a value from 1 to 6. If the new
    grid is valid call this method again to continue filling the grid.

    Parameters
    ==========
    grid_values: lst of int
      List of integers representing the symbols grid.
    """
    try:
        missing_symbol_idx = grid_values.index(0)
    except ValueError as Error:
      return grid_values

    for i in range(1,7):

      grid_values[missing_symbol_idx]=i

      if validateGrid(grid_values):
        tmp = solveGrid(grid_values)

        if tmp.count(0) == 0:
          return tmp

    return [-1]

# Get input data
input_data=input()
```

```python

# Convert string to a list of integers
grid  = list(map(int, input_data.split(" ")))

# Solve the grid
grid = solveGrid(grid)

# Return the grid formatted
print(str(grid).replace("[","").replace("]","").replace(",",""))
```

32 Curie's Waterfall

35 points

Introduction

Marie Curie, well known for her contributions to the theory of radioactivity, wants to study the behavior of a new radioactive fluid in a special environment. This environment is intended to simulate a special waterfall, represented by a grid of cells in which any of the following elements can be placed:

- Font: indicates the spot from which the fluid originates.

- Rock: a normal rock, which forces the water flow to split in two, surrounding the rock.

- Wormhole: an interdimensional teleporter which teleports the fluid flow to the other side of the wormhole.

- Black hole: absorbs the fluid, making it vanish completely.

- Magnet: modifies the path of the fluid during its fall, attracting or repelling it one unit horizontally for each cell advanced. However, the effect of the magnet is limited, and it is defined by its radius of action. It has also a power attribute, which has to be used to define the direction of the magnetic field when several magnets affect the same cell.

Are you clever enough to help Marie with her complex simulation?

IMPORTANT NOTES

- Magnets do not affect fonts, they only affect path of the fluid that flows out of the font.
- Magnets are not rocks, so they do not modify the flow of the fluid; fluid can flow through a magnet cell if needed.
- To determine if a cell is affected by a magnet or not, you have to compute the distance between the magnet and the cell, and compare that distance to the magnet's radius of action. If the distance is smaller than the radius, the cell is affected by the magnet. When more than one magnet affects the same cell, the power of all the affecting magnets must be combined (by adding or subtracting), depending on each magnet's sign (attraction or repulsion).

Input

Two integers representing the number of rows and columns of the waterfall wall, followed by a series of waterfall elements, which ends with an "end" string. Each element is represented by a different character, followed by the position of the element, represented by a row and a column. Each of the elements is defined as follows:

- Font: the character 'O'.

- Rock: the character 'R'.

- Wormhole: the character 'W' (followed by 2 connected positions).

- Black hole: the character 'B'.

- Magnet: the character 'M' followed by its position, the type of magnet (- for repulsion and + for attraction), the radius of the magnetic field, and its power (both integer).

All the given numbers are integers.

Output

A two-dimensional diagram of the waterfall, representing the path (or paths) the radioactive fluid will follow, according to the input configuration of Curie's Waterfall. Each cell is represented by a character, with an empty space represented by a dot ('.'), and the rest of the characters as outlined in the instructions.

The path of the fluid should be represented by the character 'O'.

Example 1

Input

```
20
60
O 2 7
R 4 7
M 5 7 - 4 1
R 13 5
R 13 7
R 13 9
O 4 18
R 15 17
R 16 17
R 16 18
R 16 19
R 16 20
R 16 21
R 16 22
R 16 23
R 16 24
W 16 25 4 32
R 5 32
R 10 31
M 4 38 + 7 1
B 9 38
M 10 38 + 7 1
B 15 38
M 19 29 - 5 1
O 4 44
O 4 56
M 10 50 + 15 1
M 11 50 + 15 1
R 12 50
M 18 50 - 8 3
end
```

Output

```
............................................................
............................................................
.......O....................................................
......OOO...................................................
.....O.R.O........O............OWO....M.....O...........O...
....O..M..O.......O............OR.O.........O...........O...
...O.......O......O............O...O.........O.........O....
...O.......O......O............O....O.........O.......O.....
...O.......O......O............O.....O.........O.....O......
...O.......O......O...........OOO.....B.........O...O.......
...O.......O......O...........OR.O....M..........OMO........
...O.......O......O...........O...O..............OMO........
...O.......O......O...........O....O............O.R.O.......
...O.R.R.R.O......O...........O.....O..........O.....O......
...O.......O......O...........O......O........O.......O.....
...O.......O.....ROOOOOOOO....O.......B......O.........O....
...O.......O.....RRRRRRRRW.....O............O...........O...
...O.......O....................O..........O.............O..
...O.......O.....................O........O.......M.......O.
...O.......O.................M....O........O.............O..
```

C++

```cpp
#include <iostream>
#include <string>
#include <vector>
#include <memory>
#include <cmath>

const char NOTHING    = '.';
const char FLUID      = 'O';
const char ROCK       = 'R';
const char BLACK_HOLE = 'B';
const char WORMHOLE   = 'W';
const char MAGNET     = 'M';
```

```
// Define the different types of cell

struct Cell
{
    virtual ~Cell() {}
};

struct Rock : public Cell { };

struct BlackHole : public Cell { };

struct WormHole : public Cell
{
    int rDst;
    int cDst;

    WormHole(int r, int c) : rDst(r), cDst(c) {}
};

struct Magnet : public Cell
{
    int r;
    int c;
    bool attract;
    int radius;
    int power;

    Magnet(int r, int c, bool att, int rad, int p) : r(r), c(c), attract(att), r
adius(rad), power(p) {}
};

struct Fluid : public Cell { };

// Compute the effects of the magnetic fields in the given position (r,c)
int computeMagneticField(const std::vector< Magnet >& magnets, int r, int c)
{
    int mf = 0;
    for (const Magnet& magnet : magnets)
    {
        int rDiff = r - magnet.r;
        int cDiff = c - magnet.c;
        int dist = std::sqrt(rDiff*rDiff + cDiff*cDiff);
        if (dist < magnet.radius)
        {
            if (magnet.attract)
            {
                if (cDiff > 0)
                {
                    mf -= magnet.power;
                }
                else if (cDiff < 0)
                {
                    mf += magnet.power;
                }
            }
            else
            {
                if (cDiff > 0)
                {
                    mf += magnet.power;
                }
                else if (cDiff < 0)
                {
                    mf -= magnet.power;
                }
            }
        }
    }

    if (mf == 0)
    {
        return 0;
    }
    else if (mf < 0)
    {
        return -1;
    }
    else
    {
        return 1;
    }
}

// Expand the waterfall recursively
void expandWaterfall(std::vector< std::vector< std::shared_ptr<Cell> > >& cells,
 const std::vector< Magnet >& magnets, std::vector< std::vector< bool> >& visite
d, int r, int c)
{
    if (r < 0 || r >= cells.size() || c < 0 || c >= cells[r].size() || visited[r
][c])
    {
        return;
    }
    visited[r][c] = true;

    if (std::dynamic_pointer_cast<Rock>(cells[r][c]))
    {
        if (!std::dynamic_pointer_cast<Rock>(cells[r-1][c-1]))
        {
            expandWaterfall(cells, magnets, visited, r - 1, c - 1);
        }
        if (!std::dynamic_pointer_cast<Rock>(cells[r-1][c+1]))
        {
            expandWaterfall(cells, magnets, visited, r - 1, c + 1);
        }
    }
    else if (std::dynamic_pointer_cast<BlackHole>(cells[r][c]))
    {
        // Do not do anything else, since the blackhole absorbs the Fluid
    }
    else if (std::dynamic_pointer_cast<WormHole>(cells[r][c]))
    {
        auto cell = std::dynamic_pointer_cast<WormHole>(cells[r][c]);
        expandWaterfall(cells, magnets, visited, cell->rDst + 1, cell->cDst);
    }
    else if (std::dynamic_pointer_cast<Magnet>(cells[r][c]))
    {
        expandWaterfall(cells, magnets, visited, r + 1, c);
    }
    else
    {
        cells[r][c].reset(new Fluid());

        int mf = 0;
        // Compute the magnetic filed if there is no rock below
        if (r + 1 < cells.size() && !std::dynamic_pointer_cast<Rock>(cells[r + 1
][c]))
        {
            mf = computeMagneticField(magnets, r, c);
        }
        //Ensure also that the magnetic field does not lead to a rock
        if (r + 1 < cells.size() && c + mf >= 0 && c + mf < cells[0].size() && s
td::dynamic_pointer_cast<Rock>(cells[r + 1][c + mf]))
        {
            mf = 0;
        }
        expandWaterfall(cells, magnets, visited, r + 1, c + mf);
    }
}
```

```
// Print the waterfall
void printWaterfall(const std::vector< std::vector< std::shared_ptr<Cell> > >& c
ells)
{
    for (size_t r = 0; r < cells.size(); ++r)
    {
        for (size_t c = 0; c < cells[0].size(); ++c)
        {
            if (std::dynamic_pointer_cast<Rock>(cells[r][c]))
            {
                std::cout << ROCK;
            }
            else if (std::dynamic_pointer_cast<BlackHole>(cells[r][c]))
            {
                std::cout << BLACK_HOLE;
            }
            else if (std::dynamic_pointer_cast<WormHole>(cells[r][c]))
            {
                std::cout << WORMHOLE;
            }
            else if (std::dynamic_pointer_cast<Magnet>(cells[r][c]))
            {
                std::cout << MAGNET;
            }
            else if (std::dynamic_pointer_cast<Fluid>(cells[r][c]))
            {
                std::cout << FLUID;
            }
            else
            {
                std::cout << NOTHING;
            }
        }
        std::cout << std::endl;
    }
}

int main()
{
    int rows = 0, cols = 0;
    std::cin >> rows >> cols;

    std::vector< std::vector< std::shared_ptr<Cell> > > cells(rows, std::vector<
 std::shared_ptr<Cell> >(cols));
    std::vector< Magnet > magnets;

    // Read the waterfall elements
    std::string aux;
    while (std::cin >> aux && aux != "end")
    {
        int r, c;
        std::cin >> r >> c;
        if (aux[0] == ROCK)
        {
            cells[r][c].reset(new Rock());
        }
        else if (aux[0] == BLACK_HOLE)
        {
            cells[r][c].reset(new BlackHole());
        }
        else if (aux[0] == WORMHOLE)
        {
            int rDst, cDst;
            std::cin >> rDst >> cDst;
            cells[r][c].reset(new WormHole(rDst, cDst));
            cells[rDst][cDst].reset(new WormHole(r, c));
        }
        else if (aux[0] == MAGNET)
        {
            char magnetType;
            int radius, power;
            std::cin >> magnetType >> radius >> power;
            cells[r][c].reset(new Magnet(r, c, magnetType == '+', radius, power)
);
            magnets.emplace_back(r, c, magnetType == '+', radius, power);
        }
        else if (aux[0] == FLUID)
        {
            cells[r][c].reset(new Fluid());
        }
    }

    // Traverse the waterfall grid, expanding the waterfall each time a non visi
ted Fluid cell is found
    std::vector< std::vector< bool> > visited(rows, std::vector<bool>(cols, fals
e));
    for (int r = 0; r < rows; ++r)
    {
        for (int c = 0; c < cols; ++c)
        {
            if (!visited[r][c] && std::dynamic_pointer_cast<Fluid>(cells[r][c]))
            {
                visited[r][c] = true;
                expandWaterfall(cells, magnets, visited, r + 1, c);
            }
        }
    }

    printWaterfall(cells);
}
```