CODECON

Memory Limit: 1024 MB
Time Limit: 2 s

Checkmate (750 points)

Introduction

Eve wants to play chess with Bob. But Bob is so good at chess that Eve knows that she would lose against him for sure.
Bob is such a nice friend that he suggested she can make a chess game with her own rules as long as moving rules are maintained and the goal stays as checkmate.

After some thinking she came up with these rules:
-Eve always starts first.
-Eve can choose any side of the board.
-Each player will choose one and only one figure to move on each round.
-The player can make as many moves as he wants with the same figure.
Eve wants to win the game in one round and as she likes to cheat she asks you to help her with a program to win.

Given the initial board and following Eve's rules print out:
- The side of the board to take **S** (white or black)

- The figure to choose **F**

- The minimum number of moves **M**.

You can check movement rules in https://en.wikipedia.org/wiki/Chess#Movement (https://en.wikipedia.org/wiki/Chess#Movement) .
The count of figurines for each player could vary but they must satisfy:
- count(K) = 1
- count(Q) ≤ 1
- count(N) ≤ 2
- count(B) ≤ 2
- count(R) ≤ 2

For simplicity, there will be NO pawns on the board and the king can NOT castle.

If both players can make the minimum number of moves print the white.
Priority(White) > Priority(Black)

If many figures can make the minimum number of moves for the same player print the figure with the highest priority.
Priorities are as follow:
Priority(King) > Priority(Queen) > Priority(Bishop) > Priority(Knight) > Priority(Rook)

It is guaranteed that there will be always a solution.

Input Specifications

First line contains one number **G** which is the number of games to be played.
For each game you get one line with two numbers **R**, **C** : numbers of rows and columns of the board.
Followed by R lines each with C characters describing one line of the initial board.

An empty cell will have a dot '.' The figures for the white player will be marked **K** (King), **Q** (Queen), **B** (Bishop), **N**(Knight) and **R** (Rook). The black player will have the same letters but in lower case i.e. **k**, **q**, **b**, **n** and **r.**

1 ≤ G ≤ 10
2 ≤ R,C ≤ 100

Output Specifications

For each game print in one line **S**, **F** and **M**
For example, if Eve needs to choose the white player and play 2 moves with queen for checkmate the solution would be:
W Q 2

Sample Input/Output

Input

```
1
4 5
.K..N
.....
.....
.n.k.
```

Output

```
W N 2
```

Explanation

The white knight can get to the black king in two steps

Input

```
1
4 4
k.B.
....
..q.
.K..
```

Output

```
B q 1
```

Explanation

The white bishop can get to the black king in 2 steps. However the black queen can get to the white king in 1 step.

CODECON

Memory Limit: 1024 MB
Time Limit: 2 s

Help House Targaryen ! (300 points)

Introduction

Daenerys has arrived at King's Landing to battle Cersei.

Cersei, being fiercely protective, has managed to build a gate that cannot be burned down by dragons. Instead, this gate needs a key to open.

Luckily, Daenerys has Tyrion on her side. Since we know that Tyrion "drinks and knows things", he figured out that:

- The key to the door is a binary string which does not have "**010**" in it.

- Any key of that form can open the door

- It is possible to flip a bit from **0** to **1** (and vice versa) in a single step.

Given a binary string Daenerys needs your help to find minimum steps she needs to convert the string a to valid key.

Input Specifications

The first line contains an integer, **N** (which is length of binary string **B**).

The second line contains a single binary string **B** of length **N**.

- $1 \leq N \leq 100000$
- Each character in $B \in \{$**0 , 1**$\}$

Output Specifications

Print the minimum number of steps needed to convert given binary string to key.

Sample Input/Output

Input

```
7
0101010
```

Output

```
2
```

Explanation

0101010 can be converted to 0111010 by changing the 3rd bit in 1st step. 0111010 can be converted to 0111011 by changing the 7nd bit in next step

---

Input

```
5
01100
```

Output

```
0
```

Explanation

01100 does not have any '010' so this is the key and requires no change

CODECON

Memory Limit: 1024 MB
Time Limit: 5 s

Murder at Mary Mead Hall (700 points)

Introduction

There's murder afoot at St Mary Mead Hall; The country's most dangerous mansion! Lots of nefarious characters are frequently seen creeping around the building. There are so many crimes that the police don't have enough men and women to work on them all so they've asked you to create a crime-solver.

Input Specifications

You will be given clues based around 4 criteria: - Suspect Name, Weapon, Room and Time

Each criterion is only associated with one of each other type of criterion. i.e. A suspect's fingerprints are only found on one weapon, they were only seen entering one room at one particular time. No room was entered twice. No two people were in the building simultaneously, etc. You must associate each suspect with their room, weapon and time and then wait for evidence from the forensics lab to select your prime suspect.

The first 5 lines of input are metadata in the form of:

1) The number of criterions in each list. There will be the same number of suspects as rooms, weapons and times. This number will be between 2 and 10.

Then 4 space delimited lists follow, (which will also be terminated with a space):

2) List of suspects

3) List of rooms

4) List of weapons

5) List of times (given as single integers to the nearest hour of a 24h clock time)

The metadata will be followed by an unlimited number of clues which will be in the format:

<criterion>"=="<criterion>

or

<criterion>"!="<criterion>

For example, "RIPPER==AXE" means that Ripper's fingerprints were found on the axe, while "STUDY!=16" means there was no one in the study at 16:00.

You will always be given enough clues to deduce the relationship between all criteria

Lastly you will be given a piece of forensic evidence starting with ## which will positively identify either a room, weapon or murder time from which you must deduce who the killer was.

Output Specifications

You must submit the name of the prime suspect as output

Sample Input/Output

Input

```
3
RIPPER CRIPPEN BORDEN
LIBRARY STUDY KITCHEN
AXE ACID KNIFE
14 16 18
RIPPER==AXE
RIPPER==16
CRIPPEN==18
CRIPPEN==KNIFE
ACID==KITCHEN
STUDY!=16
##ACID
```

Output

```
BORDEN
```

Explanation

from the data above, it's possible to deduce:- Borden = Acid = 14 = Kitchen, Crippen = Knife = 18 = Study, Ripper = Axe = 16 = Library Then by processing the final piece of forensic evidence, you can see that Borden is the culprit

CODECON

Memory Limit: 1024 MB
Time Limit: 5 s

Mystery Message (150 points)

Introduction

Matt manages the daily pantry treat selection. Every Monday he writes down that week's treat names on a public ledger for accounting purposes, but he doesn't want everyone to be able to tell what the treats are, otherwise there will be overcrowding.

As such, every week he picks a random number **X**, where **1 ≤ X ≤ 25**, and uses that in a Caesar cipher(https://en.wikipedia.org/wiki/Caesar_cipher (https://en.wikipedia.org/wiki/Caesar_cipher)) to encode the treat names. For example if **X = 9**, "Scones" becomes "Blxwnb". He then uses that **X** to encode that week's five treat names.

You want to know what the treat will be ahead of time, so you decide to write a program that will figure out what that week's **X** is. You wait until you find out what Monday's treat is and use that to help yourself figure out the other treats.

Input Specifications

Your program must read from STDIN:-

N lines, each containing a string that lists that week's treat names for each weekday, each name separated by a space. At the end of the line is Monday's treat, decrypted.

Output Specifications

Based on the input, print out the value of **X** for each week.

Sample Input/Output

Input

```
Zhssvaf Pbbxvrf Oernq Pubpbyngr Byvirf Muffins
```

Output

```
13
```

Explanation

We figure out that X is 13.

Input

```
Blxwnb Qdvvdb Yrn Hdv hdvb Scones
```

Output

```
9
```

Explanation

We figure out that X is 9.

CODECON

Memory Limit: 1024 MB
Time Limit: 5 s

Verifying Validity (100 points)

Introduction

Vanessa manages an online book catalog and has to manually input any new books into the system. Since she sometimes mistypes the ISBN, she would like a program to check if a given ISBN is valid.

To check the validity of an ISBNs a checksum is calculated from its digits specifically for the purpose of catching transcription errors. The checksum is calculated as the sum of all digits, each multiplied by its (integer) weight. Weights are assigned based and the position of the digit in the ISBN and alternate between 1 and 3. For example the first digit will get a weight of 1, the second will get a weight of 3, the third will get 1 again and so on.

To be a valid ISBN, the checksum must be a multiple of 10.

For example, 1234567890128 would be a valid ISBN, because 1*1 + 2*3 + 3*1 + 4*3 + 5*1 + 6*3 + 7*1 + 8*3 + 9*1 + 0*3 + 1*1 + 2*3 + 8*1 = 100 which is divisible by 10.
If the last digit was 3 instead of 8 i.e. 1234567890123, using the formula we would only get 95, which is not divisible by 10, thus this would be an invalid ISBN.

Input Specifications

Your program must read from **STDIN:-**

A single integer N ($1 \leq N \leq 10$) indicating the number of ISBN entries to verify.

N lines each containing a single ISBN (ISBNs are 13-digit integers, between 1000000000000 and 9999999999999 inclusive).

Output Specifications

Based on the input, print out either "VALID" or "NOT VALID" for each ISBN.

Sample Input/Output

Input

```
4
1234567890123
1234567890128
1111111111116
1111111111118
```

Output

```
NOT VALID
VALID
VALID
NOT VALID
```

Explanation

By applying the formula used above, we check if the ISBN is valid.

CODECON

Memory Limit: 1024 MB
Time Limit: 5 s

Heraldic Heresy (200 points)

Introduction

Lotlair decided that giving each and every team in London a coat of arms would be great for team-building.

Ever conscious of costs, he outsourced the creation of them to one consultancy he found on teletext.

To Lotlair's horror, when the coats of arms arrived, they were all wrong. As it is widely known, heraldic imagery has to observe the rule of tincture: *metal should not be put on metal, nor colour on colour*. Most of the coats of arms were breaking this rule. Now Lotlair has to send the faulty coats of arms back to the consultancy so that they can be fixed.

But there's so many of them! That's where you come in. Write a program that given a coat of arms prints if it is VALID or INVALID.

Input Specifications

Two integers, N and M, followed by N lines of M characters, representing the coat of arms. $2 < N < 100$, $2 < M < 100$.

Each character represents either a colour or a metal. There are 7 possible characters:

Metals:

O (Or, gold)

A (Argent, silver)

Colours:

a (azure, blue)

g (gules, red)

s (sable, black)

v (vert, green)

p (purpure, purple)

Consider that a colour is put on another colour if these two colours touch and they are **different**.
Assume that the colours don't touch diagonally.
Note that there may be more than one neighbouring block of the same colour/metal e.g. OOOaaa is VALID, but OAaaa or OOaag are INVALID

Output Specifications

Single word, VALID or INVALID.

Sample Input/Output

Input

```
2 2
0a
a0
```

Output

```
VALID
```

Explanation

No two different metals or two different colours are touching.

Input

```
2 2
00
av
```

Output

```
INVALID
```

Explanation

Azure is touching Vert.

Input

```
5 5
vApAv
vApAv
vApAv
vvAvv
vvvvv
```

Output

```
VALID
```

Input

```
5 5
vapav
vapav
vapav
vvavv
```

```
vvvvv
```

Output

```
INVALID
```

Input

```
9 6
aaaaaa
OOOOOO
gggggg
AAAAAA
ssssss
AAAAAA
vvvvvv
OOOOOO
pppppp
```

Output

```
VALID
```

CODECON

Memory Limit: 256 MB
Time Limit: 2 s

Funfair Tickets (350 points)

Introduction

Alice is a big fan of theme parks. She is going to visit another funfair which has an interesting ticketing system.

The funfair charges for rides in fun tokens (FT) and the prices are the following:

- One ride costs 2 FT

- One hour ticket costs 5 FT

- 6 hour ticket costs 20 FT

- One day (24 hour) ticket costs 50 FT

Each ride lasts one minute. She knows that the theme park charges in an optimal way to minimize the cost. It means that the system chooses the optimal ticket to buy taking into account all of their previous tickets purchased.

Alice wants to know how much will she be charged after each ride. Help her figure that out.

Input Specifications

The first line contains one number N ($1 \leq N \leq 10^6$) - the number of attractions rides taken by Alice.

Then N lines follow each one of them indicating time T in minutes after entry ($0 \leq T \leq 10^9$) at which Alice rode an attraction.

All times are different and given in ascending order.

Output Specifications

For each of the N lines output how much Alice was charged for that ride.

Sample Input/Output

Input

```
3
5
20
55
```

Output

```
2
2
1
```

Explanation

The first two times Alice will be charged for single ride tickets. For the 3rd ride she will be charged the remaining amount for 1 hour ticket (5 - 2 - 2 = 1).

CODECON

Memory Limit: 1024 MB
Time Limit: 3 s

Garden party (650 points)

Introduction

Skinny Pete is invited to a garden birthday party. He doesn’t really like parties too much, but heard that the birthday cake is going to be really amazing and he wouldn’t like to miss the chance to try it. There is only one little problem. There is a sprinkler system installed in the garden and by knowing his friends, there is a high chance of someone turning it on as a party prank.

Pete likes cake, but really hates getting wet. Luckily he found a sketch of the garden that has the location of the sprinklers and how far each one can sprinkle water.

*The garden looks like a rectangle that is open on one side and has the house in the opposite side.

*The cake is going to be in the house.

*The other two sides have fences so one can not enter through there, and the house does not have a back entrance.

Pete is interested to know if it is possible to enter the garden and get to the house without any risk of getting wet.

For simplicity we can think that the map of the garden is in Cartesian coordinate system.

*The garden is a rectangle that has sides parallel to the axes and having its bottom left corner at the origin (0, 0).

*The entrance to the garden is the left side and the the house is at the right side.

*Sprinklers are represented as circles with a center and a radius. Stepping anywhere inside such a circle might get you wet.

*For the purpose of this problem, and since Pete is so skinny, we can think of him as just a point travelling in the space, with real numbers as coordinates.

Input Specifications

First line of the standard input contains two space separated integers H and W, the height and the width of the garden.

Next line contains the number of sprinklers N.

After that N lines follow having three space separated **integers** each - Xi, Yi and Ri. This a description of a sprinkler as a circle with center (Xi, Yi) and radius Ri.

$1 \leq N \leq 128$

$1 \leq H, W \leq 1024$

$0 \leq Xi \leq W$

$0 \leq Yi \leq H$

$1 \leq Ri \leq 1024$

Output Specifications

Output a single line containing “**CAKE**” (without quotes) if it is possible to get to the house without getting wet and “**NO CAKE**”(without quotes) otherwise.

Sample Input/Output

Input

```
10 25
3
24 6 3
2 7 2
21 5 4
```

Output

```
CAKE
```

Input

```
50 80
11
10 24 10
69 17 14
37 14 13
3 15 12
15 6 12
62 11 12
33 11 15
16 9 11
0 29 15
54 3 10
12 45 10
```

Output

```
NO CAKE
```