miup2022

Maratona Inter-Universitária de Programação

FIRST FOUNDATION PROJECT

feedzai

dei departamento de engenharia informática

1 2 9 0 FACULDADE DE CIÊNCIAS E TECNOLOGIA UNIVERSIDADE D COIMBRA

Critical software

CROSSJOIN

PRESENT TECHNOLOGIES

miup2022

Maratona Inter-Universitária de Programação

Local Organisation:

1 2 9 0 UNIVERSIDADE D COIMBRA

Teams from the following Universities:

TÉCNICO LISBOA

UAlg FCT UNIVERSIDADE DO ALGARVE FACULDADE DE CIÊNCIAS E TECNOLOGIA

UNIVERSIDADE BEIRA INTERIOR

1 2 9 0 UNIVERSIDADE D COIMBRA

UNIVERSIDADE DE ÉVORA

UNIVERSIDADE NOVA DE LISBOA

U.PORTO

Coimbra, 15 de Outubro de 2022

CONTENTS

INFORMATION

Scientific Committee

- Alexandre Jesus, Universidade de Coimbra
- André Restivo, FEUP / Universidade do Porto
- Fábio Marques, Universidade de Aveiro
- **Filipe Araújo, Universidade de Coimbra (Coordinator)**
- João Dias, Universidade do Algarve
- Luís M. S. Russo, IST / Universidade de Lisboa
- Margarida Mamede, FCT / Universidade Nova de Lisboa
- Mário Pereira, FCT / Universidade Nova de Lisboa
- Pedro Mariano, ISCTE
- Pedro Ribeiro, FCUP / Universidade do Porto
- Rui Maranhão, FEUP / Universidade do Porto
- Rui Mendes, Universidade do Minho
- Simão Melo de Sousa, Universidade da Beira Interior
- Vasco Pedro, Universidade de Évora

Local Organisation Committee

- Alexandre Jesus, Universidade de Coimbra
- Amílcar Cardoso, Universidade de Coimbra
- Carlos Fonseca, Universidade de Coimbra
- Filipe Araújo, Universidade de Coimbra
- **Luís Paquete, Universidade de Coimbra (Coordinator)**

Languages and Compilers

I. Files `*.c` are assumed to be C code and are compiled with `gcc 11.2.0` with the command

```
gcc -std=gnu17 -O2 file.c -lm
```

and then executed with the command

```
./a.out
```

II. Files `*.cpp` are assumed to be C++ code and are compiled with `g++ 11.2.0` with the command

```
g++ -std=gnu++17 -O2 file.cpp -lm
```

and then executed with the command

```
./a.out
```

III. Files `*.java` are assumed to be Java code and are compiled with `OpenJDK 17.0.4` with the command

```
javac -encoding utf8 file.java
```

and then executed with the command

```
java -Xss128m -Xms512m -Xmx512m file
```

IV. Files `*.py` are assumed to be Python 3 and are executed with `Python 3.8.13 (PyPy 7.3.9)`. This version performs just-in-time compilation, with the command:

```
pypy3 -m py_compile file.py
```

and executes the code with the command

```
pypy3 file.py
```

Limits

Attention: The use of pragmas is explicitly prohibited in C/C++ solutions (such as modifying optimization level). Solutions containing pragmas will be considered invalid.

Compilation: The compilation process can take at most 60 seconds and the maximum source code size is 100 KB. Every program/solution must be submitted in a **single file**. For Java submissions, the file must have the same name as the class that contains the main method. There is no limit to the number of classes to be contained in that file.

Runtime The maximum runtime for each problem is 2 seconds, except in problem G – Doubled Secrets, which has a 1-second limit.

Input/Output

All programs should read the input from the standard input, and write the output to the standard output. All lines (both in the input and output) should end with the newline character (`\n`). Except when explicitly stated, a single space is used as a separator. No line starts or ends with any kind of white space.

Development Environment

- Operating System
  - Ubuntu 22.04
- Graphical Environment
  - Gnome 3
- Text Editors
  - vi/vim/gvim
  - Emacs
  - Gedit
  - Geany
  - VSCode (with C, C++, Python and Java extensions)
- Integrated Development Environments:
  - Codeblocks
  - Eclipse
  - Spyder
  - IDEA Community Edition
  - PyCharm Community Edition
- Debuggers
  - gdb
  - jdb
  - pdb
  - valgrind
- Browsers
  - Firefox
  - Chromium
- Documentation
  - Devdocs available on Mooshak

PROBLEMS

Problem A: ~~No~~ Country for PhD and Engineers

Alice Famil, the president of the public research funding agency of the *Country for PhD and Engineers*, has made a call for more PhD and engineers among the workforce. Bob Famil, the president of the state owned company *WeCare* is going to answer the call and hire PhD and engineers. Job applications will be selected taking into account the number of followers an applicant has on the music social network *I Sing, therefore I Think*.

There are no financial constraints, therefore the company is going to hire as many PhD and engineers as possible. However, Bob Famil is afraid that followers of his future employers will start buying his company products and that production capacity cannot fulfil demand. As such, the total number of followers cannot exceed *WeCare* production capacity.

Task

Write a program that receives the production capacity of *WeCare*, followed by a list of applications by ascending order of submission. Each application contains the name, number of followers, and CV score. Your program should print the list of hired applicants alongside total number of followers and CV score.

If there is a tie between two groups of candidates, the one with the total highest CV score wins. If there is a tie on the CV score, the group with the earliest application wins. If there is tie on the earliest, then it is broken on the second most earlier, and so forth.

Input

The first line contains *WeCare* production capacity, C. The second line contains the number of applications, N. The following N lines contain application data: a string of lowercase letters representing the name, m_a, an integer representing number of followers, f_a, and an integer representing CV score, s_a.

Constraints

$1 \leq C \leq 2^{31}$	production capacity
$1 \leq N \leq 20$	number of applications
$1 \leq \operatorname{len} m_a \leq 10$	application name length
$0 \leq f_a \leq 2^{20}$	number of followers
$1 \leq s_a \leq 1000$	CV score

Output

The first line of the output should contain the number of hired applicants, the total number of followers (of hired applicants), and the total CV score (of hired applicants). Each of the following lines, should contain the applicant name, by arrival order.

Sample Input

```
18010
6
anali 5000 1
bruna 7000 2
carlo 6000 3
david 6000 2
elsay 7000 1
felip 5000 3
```

Sample Output

```
3 18000 8
bruna
carlo
felip
```

Sample Explanation

If more than three applicants are hired, their total number of followers exceed production capacity. This combination has the highest possible CV score. `bruna` appears instead of `david` because she applied earlier.

Sample Input

```
18000
6
anali 5000 3
bruna 5000 3
carlo 6000 3
david 6000 3
elsay 7000 3
felip 7000 3
```

Sample Output

```
3 16000 9
anali
bruna
carlo
```

Sample Explanation

If more than three applicants are hired, their total number of followers exceeds production capacity. Any combination of three applications has the same CV score of 9. This combination has the three earliest applications.

Problem B: Rescue by Rail

The authorities of Alertland, a volcanic island, are always concerned with the security of the people. One of the major issues is the movement of inhabitants to the only safe region on the island in case of a massive volcanic eruption.

A solution the government started studying lies in transporting everybody by train. The island is divided into small regions, each with a train station quickly reachable on foot, by bicycle or by car. The idea is that all people in an unsafe region and those arriving by train from another region would take a train that would transport them to a new region. Although the data collected so far is not enough to take into account all the details, two figures for each region have already been estimated:

- the *population size*;
- the *departure capacity*, which is the maximum number of people that can catch a train at the train station of the region.

Notice that, for every unsafe region, not only the population in the region but also all people in transit (those arriving by train from another region) should catch a train at the region train station.

On Alertland, trains are truly modular because, instead of being pulled by locomotives, each carriage has its own engine. As a consequence, if there are direct links from a region X to k distinct regions, any distribution of passengers among these k lines is viable provided the total number of people carried does not exceed the departure capacity of X.

Can you verify that this rescue by rail allows the entire population to end up in the safe region?

Task

Write a program that, given the railway network (stations and direct links), the population size and the departure capacity of each region, and the safe region, computes the maximum amount of population that can reach the safe region.

Input

The input first line has two integers: R, which is the number of regions, and L, which is the number of direct links (by rail) between regions. Regions are identified by integers, ranging from 1 to R.

R lines follow. The i^{th} of these lines contains two integers, p_i and d_i, which are the population size and the departure capacity of region i, respectively (for every $i = 1, \ldots, R$).

Each of the following L lines has two distinct integers, r_1 and r_2, which indicate that there is a two-way direct link between regions r_1 and r_2.

The last line has a single integer, S, which is the safe region.

Constraints

$2 \leq R \leq 1\,000$	Number of regions
$1 \leq L \leq 5\,000$	Number of direct links between regions
$1 \leq p_i \leq 10\,000$	Population size of region i (for $i = 1, 2, \ldots, R$)
$1 \leq d_i \leq 300\,000$	Departure capacity of region i (for $i = 1, 2, \ldots, R$)

Output

The output consists of a single line with an integer, representing the maximum amount of population that can end up in the safe region.

Sample Input 1

```
5 6
1000 1050
5000 6000
350 320
2100 2100
780 900
2 4
2 3
2 5
4 1
5 3
1 5
4
```

Sample Output 1

```
9150
```

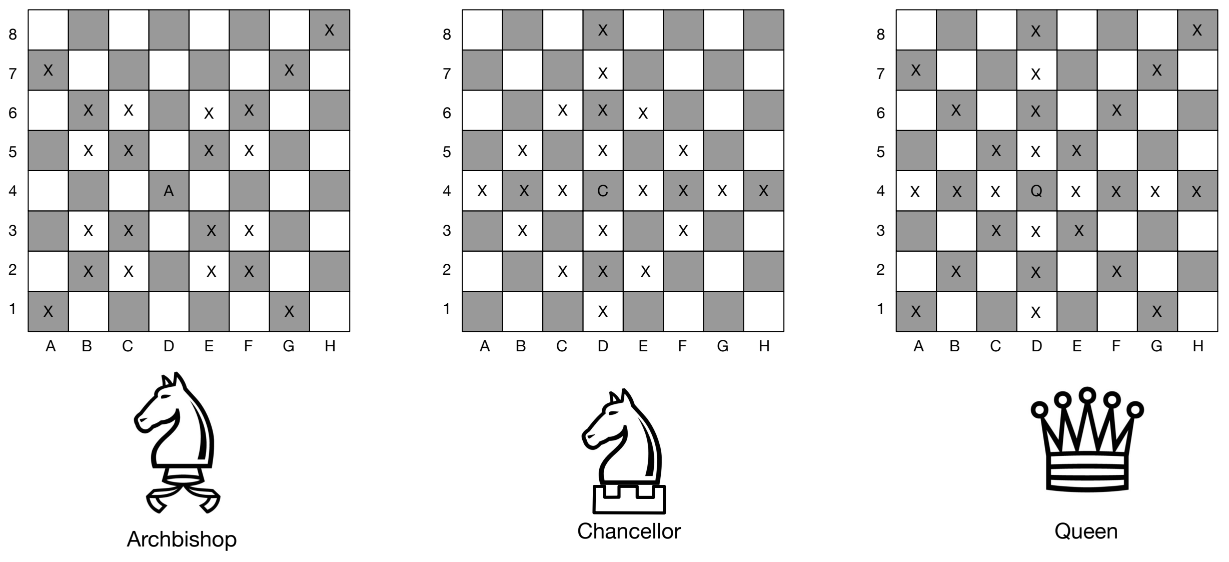
Sample Input 2

```
5 5
1000 1050
5000 6000
350 320
2100 2100
780 900
2 4
2 5
4 1
5 3
1 5
4
```

Sample Output 2

```
9000
```

Problem C: Walk Like an Archbishop or a Chancellor or a Queen



Pieces taken and adapted from vecteezy.com

An Archbishop is an unorthodox chess piece that combines the moves of the Bishop and the Knight. It was invented by the Chess World Champion José Raúl Capablanca. The movement of the Knight can be described as an "L", moving 1 or 2 squares in one direction (rows or files) and then 2 or 1, respectively, in a perpendicular direction (along a row if it first moved along a file and vice-versa). A Bishop can move along diagonals. The Archbishop can move in both ways.

Another unorthodox piece is the Chancellor, which combines the moves of the Rook and the Knight. It was invented by Benjamin R. Foster. A Rook can move orthogonally, along rows and files.

Finally, we have the Queen, a standard piece that combines the moves of the Rook and the Bishop, moving orthogonally and diagonally.

One thing we would like to know is which of the three pieces is stronger, the Chancellor, the Archbishop, or the Queen. To answer this question, you should solve a very simple challenge.

Task

Given a set of board squares, your task is to tell which of the pieces can cover all those squares, in any order, using the least number of movements. You should also compute this latter number.

Input

The input starts with a line with the number of test cases, n. Each test case starts with a line with the number of squares to visit, s, and the starting square for the piece. The following s lines contain the squares to visit. The following test case, if any, starts on the following line.

The names for the squares follow standard chess conventions: rows are marked with a number (1 to 8), and files with a capital letter ("A" to "H").

Constraints

$1 \leq n \leq 5$	Number of test cases to evaluate
$1 \leq s \leq 10$	Number of squares to visit

Output

For each of the n test cases, you should output the minimum number of moves needed to cover the required squares, when starting from s, followed by a space and by a letter denoting the best piece for the tour ('A' for Archbishop, 'C' for Chancellor, 'Q' for Queen). In case of a tie, you should write the letters corresponding to all the tied pieces, in alphabetic order, with no spaces between them.

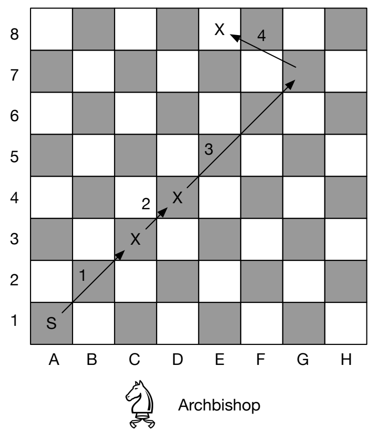
Sample Input 1

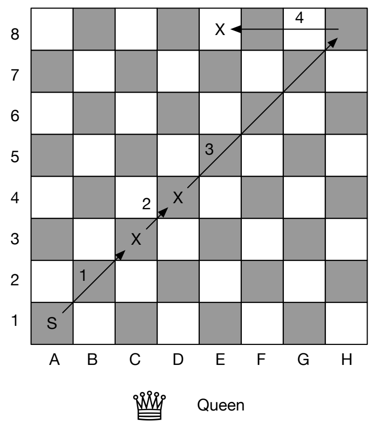
```
1
3 A1
C3
D4
E8
```

Sample Output 1

```
4 AQ
```

For this specific input, the following image provides examples of how to cover all the `x`-marked squares starting from A1 (marked with the `S`), with an Archbishop and a Queen, in 4 moves:



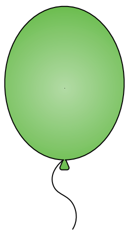


Sample Input 2

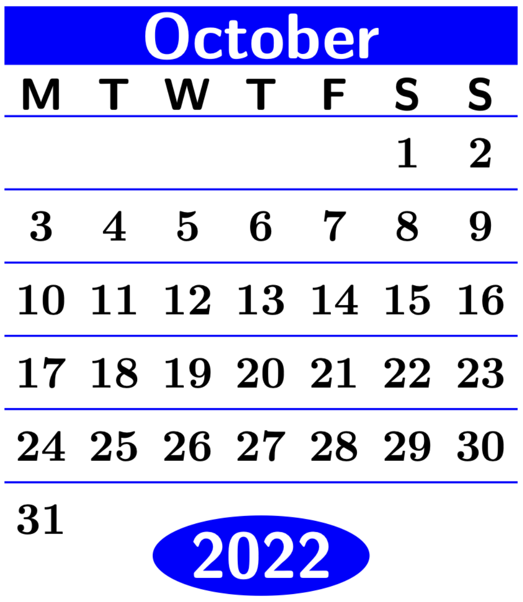
```
2
3 A1
C3
D4
E8
4 C5
C2
A1
H8
B3
```

Sample Output 2

```
4 AQ
4 A
```



Problem D: Weekday Calculator



October

M	T	W	T	F	S	S
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						

2022

This folklore rhyme may help you remember the number of days in a month:

Thirty days has September,
April, June and November.
All the rest have thirty-one,
Saving February alone,
Which has twenty-eight, rain or shine.
And on leap years, twenty-nine.

Our goal is to determine which day of the week corresponds to a certain date. The following information might be useful, but you may use any prior knowledge you may have:

- Each week has seven days: Monday, Tuesday, Wednesday, Thursday, Friday, Saturday and Sunday.
- The 1st of January of 1900 was a Monday.
- Months have a different amount of days, described by the rhyme above.
- A leap year occurs on any year evenly divisible by 4, but not on a century unless it is divisible by 400.

Note that according to the rules above February 1900 had 28 days and **not** 29.

Task

Develop a tool that receives a date and returns the corresponding day of the week. Your tool should be as fast as possible!

Input

A sequence of N lines containing a day d a month m and a year y.

Constraints

$1 \leq N \leq 10^4$	Number of dates
$d \in \{1, \ldots, 31\}$	day
$1900 \leq y \leq 5000000$	year
$m \in \{$ `Jan`, `Feb`, `Mar`, `Apr`, `May`, `Jun`, `Jul`, `Aug`, `Sep`, `Oct`, `Nov`, `Dec` $\}$.	month

Output

The output consists of a sequence of N lines, i.e., the same number as the input. Each line should contain the corresponding weekday followed by a newline.

Sample Input

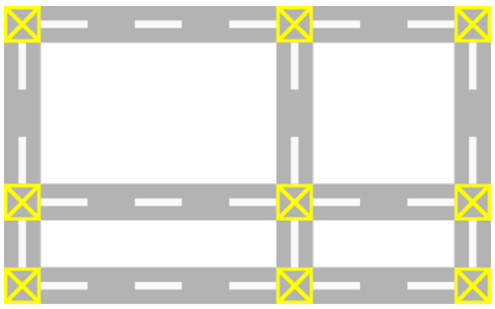
```
1 Jan 1900
28 Jul 1914
24 Feb 4999999
25 Apr 1974
1 Sep 1939
26 Apr 1986
2 Sep 1945
```

Sample Output

```
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
```

Problem E: Meeting with a friend

Ana and Pedro live in a city where the streets are laid out in a grid, as exemplified by the following figure:



Intersections between the streets, shown in yellow in the figure above, are given by pairs of integer coordinates (X_i, Y_j), $1 \leq i \leq N$, $1 \leq j \leq M$, such that $X_i < X_{i+1}$ and $Y_j < Y_{j+1}$. The shortest distance between any two intersections is given by the Manhattan distance.

Ana is currently at the intersection with coordinates (X_1, Y_1), and Pedro is at the intersection with coordinates (X_N, Y_M). They want to meet each other, but want to choose an intersection that is at a similar distance to both of them, so that they both walk a similar distance. Otherwise, the person that walks more might be upset.

Task

Your task is to help Ana and Pedro find an intersection such that the absolute difference between the shortest distances traveled by both Ana and Pedro to get to that intersection is minimal.

Input

The first line gives the number of test cases T. Each test case consists of three lines. The first line of a test case gives two integer N, M. The second line gives N integers corresponding to the values of $X_1, \ldots, X_N$ in order. The third line gives M integers corresponding to the values of $Y_1, \ldots, Y_M$ in order.

Constraints

$1 \leq T \leq 100$
$1 \leq N, M \leq 2 \times 10^5$
$1 \leq X_i, Y_j \leq 10^8$

The sum of N and the sum of M over all test cases do not exceed 2×10^5.

Output

For each test case print a line with two integers separated by a space that correspond to the X and Y coordinates of the intersection where Ana and Pedro will meet. If there are multiple intersections that give a minimal absolute difference, the lexicographically smallest should be printed.

Sample Input

```
2
4 3
1 7 10 14
1 4 6
5 7
28 30 43 60 92
20 29 49 51 59 92 94
```

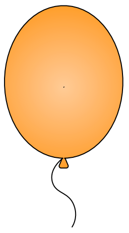
Sample Output

```
7 4
60 59
```

Sample Explanation

In the first test case, the intersection at coordinate $(7, 4)$ is at distance 9 from both Ana, who starts at $(1, 1)$, and Pedro, who starts at $(14, 6)$. Therefore, the absolute difference between the distances travelled by both of them is 0, which is obviously minimal. The intersection at $(10, 1)$ also gives an absolute difference of 0. However, $(7, 4)$ is lexicographically smaller.

For the second test case, the intersection $(60, 59)$ is at distance 71 from Ana, who starts at $(28, 20)$, and distance 67 from Pedro, who starts at $(92, 94)$. Therefore, the absolute difference between the distances is 4, which is minimal for this test case. In this case all other intersections give a larger absolute difference.



Problem F: Was it a Dream?

You find yourself in a cramped, spherical space. You try to move, but other than feeling a small impact, nothing happens. You then try to move in the opposite direction, and you feel yourself start rolling. You roll, roll, roll, then you bump into something and you come to a stop. Strangely enough, you are okay, albeit a bit sick. Having nothing better to do, you keep at it for some time. At some point, you realise that you have some control over the direction of the rolling and that, once you start rolling, you only stop when you hit something that must be some kind of obstacle. After that, you may only move in the direction you came from, or at a right angle from it. Suddenly, just as you begin getting tired of what seems like a pointless exercise, your stomach reaches for your mouth as you run out of ground and start falling. And you keep falling. What have you gotten yourself into now? Then, there's a crash. The sphere shatters. You're home.

Task

Given the map of the rectangular space where the sphere moves in, and your initial location, your task is to compute the minimum number of moves it takes to reach the hole which will take you home (it is not so much the rolling that upsets you, but the bumping into things). Each location of the space is either empty, or it contains an obstacle or the hole.

The sphere may only move through the empty locations, in the directions parallel to the sides of the map. Once the sphere starts moving, it only stops when it hits an obstacle or it falls through the hole. If the sphere falls off a side of the map, there is no way of getting back on the map again and you will never get home.

Input

The first line of input contains three integers, R, C and T, representing, respectively, the number of rows and columns of the map, and the number of test cases.

Each of the following R lines contains C characters, describing a line of the map. An empty location is represented by the character '`.`' (dot), an obstacle is represented by the letter '`O`', and the single hole is represented by '`H`'.

The final T lines contain a pair (r_i, c_i) of integers each, which represents your initial location in the i^{th} test case. The initial location is always empty. Location $(1, 1)$ corresponds to the first character of the first row, in input order, and location (R, C) corresponds to the last character of the last row.

Constraints

$1 \leq R, C \leq 900$	Number of rows and of columns of the map
$1 \leq T \leq 10$	Number of test cases
$(1, 1) \leq (r_i, c_i) \leq (R, C)$	Initial locations

Output

The output consists of one line for each test case, containing either a single integer, denoting the minimum number of moves it takes you to reach the hole, starting from the corresponding initial location, or the word `Stuck`, if it is not possible to reach the hole from the initial location.

Sample Input

```
10 20 3
.....O..............
OOO.OO.OOOOOOOO.....
O.............O..OOO
O.O..O.............O
O.O..O........OOO..O
..O..O..........O..O
..O..OOOO.......O..O
..O..........H.....O
..O................O
..OOOOOOOOOOOOOOOOOO
8 1
8 5
1 10
```

Sample Output

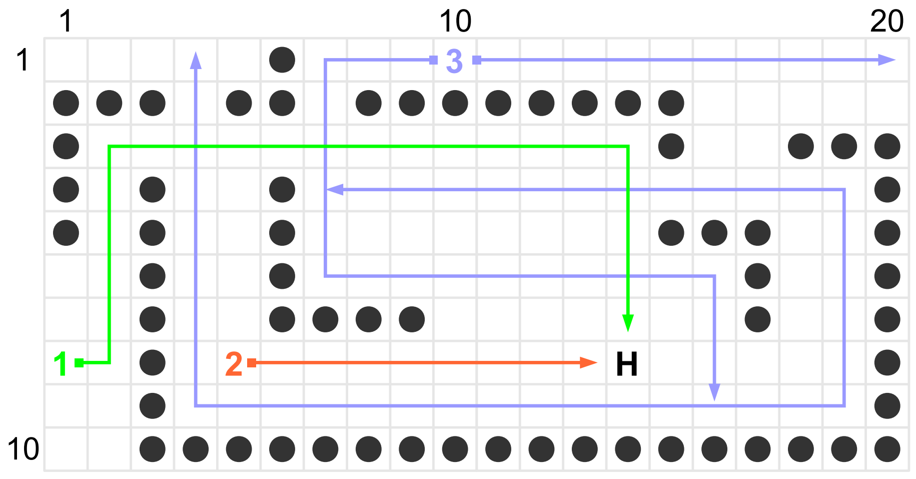
```
4
1
Stuck
```

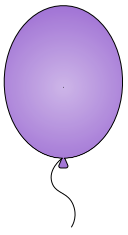
Sample Explanation

From location $(8, 1)$ (where the green 1 is), you can move right, stopping at location $(8, 2)$ due to the obstacle in location $(8, 3)$. From there, you can move down, falling off the map, or up, stopping at $(3, 2)$. From $(3, 2)$, you can move right, to $(3, 14)$, and then down, falling through the hole when you reach location $(8, 14)$.

At location $(8, 5)$ (the orange 2), you are in line of sight of the hole and you only have to move right to reach it and get home.

If you start from location $(1, 10)$ (the blue 3), you may only stop at locations $(1, 7)$, $(6, 7)$, $(6, 16)$, $(9, 16)$, $(9, 4)$, $(9, 19)$, $(4, 19)$, and $(4, 7)$, neither of which gives you access to the hole.





Problem G: Doubled Secrets

Hänsel and Gretel have been captured by the witch. While they are being fattened, to be served as the main course of a delicious dinner, the witch keeps them apart. The only means they have found to communicate with each other and devise a way to escape is through written messages, left in the bathroom, which they are allowed to visit in turn.

They do not want the witch to be able to read their messages and find out what they are plotting. Being kids, advanced maths is not yet part of their curricula and, in their last moments together, before being separated, they came up with a simple scheme to encipher the messages in three steps:

1. They remove every space, diacritic mark and punctuation symbol from the message, keeping only the letters, which they convert to lower case.
2. They append to the resulting string a reversed copy of itself.
3. Finally, they insert random sequences of letters in random places of the string obtained in the second step.

If the final step is not done with care, the insertion of random letters may result in adding new letters to the message, and they know that they will have to be extra careful, to ensure that that does not happen.

They also realised that sometimes there could be some ambiguity in the deciphering of the message. To remove any doubt about which is the next character of the message, they decided to build the ciphered text in such a way that, as long as there duplicate characters in the ciphered text, there will always be a pair of occurrences of one character whose distance is greater than the distance between any other two remaining occurrences of any character, and that character is the next message character.

Task

Your task is to retrieve the original message from the enciphered text.

Input

The input consists of one line, with L lower case letters.

Constraints

$2 \leq L \leq 20\,000$ Length of the cipher

Output

The output will consist of a single line, with the original message, which is never empty.

Sample Input

```
wtreqbhcsatziwkkyniqsgdxgjnitsrxjyefta
```

Sample Output

```
testing
```

Problem H: No More Stacking

In Formula 1 (F1), the main discipline in the car-racing world, teams of two drivers line up to fight for the very prestigious titles of World Driver Champion and Constructors Champion. In a such competitive sport, teams employ as many resources as they can in order to come up with the best strategy for each race. These include sophisticated simulators, live data gathering of tire degradation, as well as fuel consumption optimization.

One key aspect that F1 teams must take into serious consideration is how they manage their pit-stop strategies. Changing tires at the right time can play a crucial role in search for victory. However, a much simpler aspect is to be taken into consideration: each team only has a single box slot, so never pit the two drivers at the same time! This would mean that one of the drivers would have to wait completely stationary behind his team mate, losing crucial racing time. This is called *stacking* in the F1 jargon.

F1 is experiencing crazy times: in order to make the sport even more popular, stewards are now allowing teams to run *more than two* cars. But teams keep their **single** box slot, hence they are desperate to figure out how to avoid stacking between their drivers.

You have just been hired as the computer science expert for a F1 team, and your first task could not be clearer: given possible pitting strategies, you must predict any possible stacking scenario. In this new era of F1, drivers are very well-disciplined and cars are very reliable. This means: (1) there are no overtakes between team-mates (*i.e.*, if driver i starts the race ahead of driver j, the former will finish the race before the latter); (2) the cars never break and no component is ever damaged (*i.e.*, no need for a pit-stop to replace parts of the car, only about tires-changing strategy); (3) finally, same team drivers keep enough space between each other so you only need to worry about stacking of two consecutive cars (*i.e.*, there is enough space between car k and $k+i$, with $i > 1$ so that they will never arrive at the same time to the pit box).

Prior to the start of the race, manufacturers provide information about which tires are available for the race, as well as the *stint* of each class of tires. A stint is simply the number of laps elapsed between each pit-stop. In other words, the maximum number of laps a certain class of tires lasts before being changed. Since F1 stewards are deeply concerned about the planet, once a set of tires is put into the car, they must be used to its full extent. This includes the stint between the last pit-stop and race ending. For instance, if for a 8-laps race and two drivers you are given soft tires lasting two laps and hard tires lasting four, a possible race strategy is as follows: for the first pilot, use a set of hard tires (four laps), make a pit-stop, use another set of hard tires (another four laps); for the second pilot, use a set of soft-tires (two laps), make a first pit-stop and put on hard tires (four laps), make a final pit-stop and use soft tires for the final two laps. However, assigning a soft-soft-hard strategy to your second driver would result in a stack situation. The strategy soft-hard-hard is invalid, as well, since this would make the last stint of hard tires to last only for two laps, while this kind of tires must last exactly four laps.

Task

Your task is simple: given the number of laps in the race, the number of drivers in the team, the description of the available set of tires and their duration stint, can you compute the number of non-stacking race strategies for your team? Consider you have unlimited supply for each set of tires.

Input

The input consists of a first line with an integer N which stands for the number of laps. The following line consists of an integer D, standing for the number of drivers of the team. The third line is an integer T, which stands for the number of available tire classes. The following T lines represent each class of tires, as a single integer S. Here, S is the stint of the tires. No two different classes of tires can have the same stint.

Constraints

$5 \leq N \leq 50$	Number of laps
$0 < D \leq 20$	Number of drivers
$1 < T \leq 5$	Number of tire classes
$1 < S_i \leq N$	Stint of tires from class i

Output

A single line with an integer representing the number of possible non-stacking strategies. You can assume there are always less than 2^{63} such strategies.

Sample Input 1

```
8
2
2
2
4
```

Sample Output 1

```
2
```

Sample Input 2

```
9
3
2
2
3
```

Sample Output 2

```
8
```

Input/Output 2 Explanation

Let us explain in detail the result one gets in the second example above. For such case, the F1 team faces a 9 laps-race with 3 drivers. Tire manufacturers have provided tires that last two laps (let us call them *soft, S* tires) and others that last 3 laps (let us call them *intermediate, I* tires).

All valid race strategies must use a tires set to its full stint. Hence, for instance, equipping one of the drivers with soft-intermediate-soft-intermediate is not acceptable since it would use the last intermediate set only for two laps, instead of the expected three.

An example of a valid strategy for these conditions is as follows:

- First driver equipped with S-I-S-S.
- Second driver equipped with I-I-I.
- Third driver equipped with S-I-S-S. Remember that the first and third drivers will never get to the box at the same time. Henceforth, even if both start the race in soft tires and make their first pit-stop after 2 laps, this is still a valid strategy.

The remaining 7 valid strategies are the following ones:

Driver 1	Driver 2	Driver 3
I-I-I	S-I-S-S	I-I-I
I-I-I	S-S-I-S	I-I-I
I-S-S-S	S-S-S-I	I-S-S-S
S-S-I-S	I-I-I	S-I-S-S
S-I-S-S	I-I-I	S-S-I-S
S-S-I-S	I-I-I	S-S-I-S
S-S-S-I	I-S-S-S	S-S-S-I

Problem I: Sprinkling Sprinklers

When George first received the news, he was overjoyed. It isn't every day that you discover you've inherited a large piece of land from an uncle you never even knew existed. He decided to become a proper farmer; he sold everything, packed, and took the first train out of town.

When he arrived at the farm, he was met with a big surprise. The farm was huge, just as promised, but it was oddly shaped. It was as long as the eyes could see but very narrow, only a few meters wide; it was also ridden with obstacles (see Figure 1). George's enthusiasm quickly turned to discouragement. He thought about giving up and returning home, but he knew he would be admitting defeat. Instead, he decided to stay and try to make the best of it.

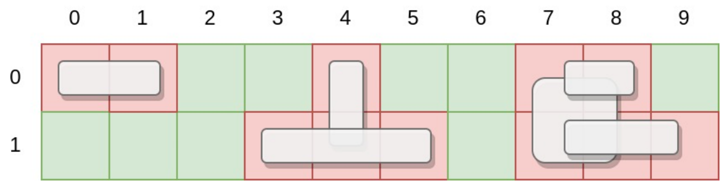


Figure 1: An example of a possible farm. Each cell represents a square meter. Some cells are not arable due to obstacles; some obstacles overlap.

Due to the odd shape, a big irrigation system was out of the question, so he decided to install a small sprinkler on every square meter of arable land (see Figure 2).

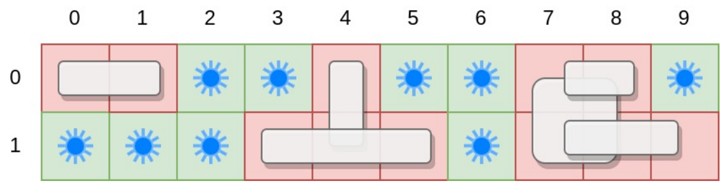


Figure 2: Putting a sprinkler on every arable square meter.

Task

Your task is to calculate how many sprinklers George has to buy to fill every square meter of arable land. The farm is a perfect rectangle with a certain length and width and is divided into $1\,\mathrm{m} \times 1\,\mathrm{m}$ cells. Obstacles can overlap and occupy several adjacent cells, always forming a rectangle of non-arable land.

Input

The first line contains the field's length L and width W, and the number of obstacles O. Each of the remaining O lines represents an obstacle (in no particular order). Each of these lines will contain the coordinates of the cells occupied by the obstacle, given by the top-left corner, $X1$ and $Y1$, and the bottom-right corner, $X2$ and $Y2$.

Constraints

$1 \leq L \leq 1000000$
$1 \leq W \leq 100$
$0 \leq O \leq 20000$
$0 \leq X1 \leq X2 < L$
$0 \leq Y1 \leq Y2 < W$

Output

A line with a single integer representing the number of sprinklers to install.

Sample Input

```
10 2 6
0 0 1 0
7 0 8 0
4 0 4 1
7 1 9 1
3 1 5 1
7 0 8 1
```

Sample Output

```
9
```

Sample Explanation

A sample input representing the same field as depicted in Figure 1. The field's length is 10, the height is 2, and there are 6 obstacles. The 9 sprinklers can be seen in Figure 2.

Problem J: Climate Change

The EOF (Environmental Organization Fund) is sponsoring a new study on climate change. Among many other datasets, they have a sequence of N integer numbers representing the daily (average) temperature over the last N days on any given location (since they are using the new universal temperature metric, these integers can get very large).

Researchers are always keen on reporting that a certain day is really hot, but they want to be sure of how it compares to previous temperatures recorded. In particular, given a range $[a, b]$ of consecutive days, they would really like to know in how many days was the temperature above a certain value k. For instance, imagine we have registered the following temperatures over the last $N = 10$ days:

Day	1	2	3	4	5	6	7	8	9	10
Temperature	32	31	30	25	27	28	35	32	32	31

If we wanted to know the amount of days with temperature above $k = 30$ between days $a = 1$ and $b = 9$, the answer would be 5 (days 1, 2, 7, 8 and 9). If however the query is days above $k = 32$ between $a = 8$ and $b = 10$ then the answer would be zero.

Unfortunately, the researchers found that some values were wrong and so they want a way to change the temperature of any given day d to a new value t and continue doing queries. For instance, given the previous example, if we change day $d = 8$ to $t = 33$, the number of days above $k = 32$ between $a = 8$ and $b = 10$ now becomes 1.

Task

Given an initial sequence of N integers representing daily temperatures, your task is to implement two operations: (i) determine number of days with temperature $> k$ within a given range $[a, b]$ of days; (ii) change the value of day d to temperature t, affecting the outcome of all operations that follow.

Input

The first line contains N, the number of days to consider. The second line contains N space separated integers v_i representing the sequence of initially recorded temperature values.

The third line contains M, the number of operations to consider. The following M lines contain the description of the operations to apply, one per line. A query operation is given by four integers "1 a b k", and a value change operation is given by three integers "2 d t", as previously explained. It is guaranteed that at least 50% of the operations are queries.

Constraints

$1 \le N \le 75\,000$	Number of days
$1 \le M \le 75\,000$	Number of operations
$-10^8 \le v_i, t, k \le 10^8$	Temperature values
$1 \le a \le b \le N$	Range of days
$1 \le d \le N$	Day to change temperature

Output

One line with a single integer for each querying operation, indicating the amount of days within range $[a, b]$ with temperatures bigger than k. Note that the result should reflect any temperature changes applied before.

Sample Input

```
10
32 31 30 25 27 28 35 32 32 31
8
1 1 9 30
1 8 10 32
2 8 33
1 8 10 32
2 10 40
1 8 10 32
2 1 -1
1 1 10 -1
```

Sample Output

```
5
0
1
2
9
```

Problem K: Strike

The labor union of a Portuguese ground handling company, called *Portforce*, operating in the airports of Lisbon, Porto, Faro, Funchal, and Beja, is planning a one-day strike to claim better wages and better work conditions. To have the highest possible impact, the labor union wants to select a day in **November** for the strike that will maximize the number of canceled flights among all airports where they operate.

Flights will be canceled if, on the day of the strike[1], the plane is flying from or to an airport where the handling company operates and the airline operating the flight has a handling contract with *Portforce*. For example, a flight departing from Munich (an airport not serviced by *Portforce*) at 22:00 on the day of the strike, and arriving at Lisbon at 01:00 on the following day (an airport serviced by *Portforce*), will not be canceled. Also, the strike will not cancel flights if the airline responsible for the flight does not have a contract with *Portforce*.

Task

Develop a program that receives information on airlines that have a handling contract with *Portforce*, and a list of flights spanning several days in November, and determine the day when a strike would maximize the number of canceled flights across the airports of Lisbon, Porto, Faro, Funchal, and Beja. In case of a tie (*i.e.*, two or more days have the same number of canceled flights), you should select the earliest day. The program should also calculate the total number of flights canceled.

Input

The first line contains N, the number of airlines with a handling contract with *Portforce*. The second line contains a sequence of N, space-separated strings of letters, each corresponding to an abbreviated name of an airline with a handling contract with *Portforce*.

The third line contains M, the number of flights to consider. The following M lines will correspond to a sequence of flights, one per line. The following space-separated fields

[1]The strike takes place between 00:00 and 23:59 of the selected day.

represent a flight: *Airline* (the name of the airline), $FromAirport$ (the name of the departing airport), $FromDay$ (the departure day), $FromHour$ (the hour the flight is departing), $ToAirport$ (the name of the arriving airport), $ToDay$ (the arriving day), and $ToHour$ (the hour the flight is arriving).

Airline, $FromAirport$, and $ToAirport$ are all strings containing only letters. $FromDay$ and $ToDay$ are integers in the range [1,30] representing the day of the departure/arrival in November. $FromHour$ and $ToHour$ represent a time using the *hh:mm* format.

Airports where *Portforce* operates, will always be written and capitalized as Lisbon, Porto, Faro, Funchal, and Beja. For two airlines to be considered the same, their strings must be exactly the same (including capitalization). All names (airports and airlines) have at most 10 characters ([a-zA-Z]).

Constraints

$1 \leq N \leq 20$	Number of airlines working with *Portforce*
$1 \leq M \leq 10000$	Number of flights in November
$1 \leq d \leq 30$	Range of different days to consider
$ToDay - FromDay \in \{0, 1\}$	Days must be consecutive
$FromHour < ToHour \vee ToDay \neq FromDay$	We cannot go back in time

Output

Two lines. The first line contains the first strike day that maximizes the number of flights canceled. The second line has the total number of flights canceled if that day is selected.

Sample Input 1

```
3
TAP Vueling Lufthansa
7
TAP Porto 21 23:10 Faro 22 00:25
Easyjet Orly 21 14:00 Lisbon 21 15:30
TAP Lisbon 22 09:00 Faro 22 09:45
Lufthansa Berlin 21 17:00 Porto 21 20:30
Vueling Lisbon 23 07:35 Sevilla 23 08:35
Lufthansa Faro 22 11:10 Munich 22 13:20
Lufthansa Munich 21 06:55 Stockholm 21 09:03
```

Sample Output 2

```
22
3
```

Sample Input 2

```
1
TAP
2
TAP Munich 7 23:10 Paris 8 00:25
Vueling Porto 12 12:25 Lisbon 12 13:10
```

Sample Output 2

```
1
0
```

Sample Explanation

In the first sample input, the first flight will be canceled if days 21 or 22 are selected (both departure and arrival are in airports of interest on different dates), so it should be canceled for both days. The second flight is ignored since *Easyjet* flights are not handled by *Portforce*. The last flight is also ignored because neither airports are relevant. In total there are 2 canceled flights in day 21 (*TAP* Porto–Faro, *Lufthansa* Berlin–Porto), 3 canceled flights in day 22 (*TAP* Porto–Faro, *TAP* Lisbon–Faro, *Lufthansa* Faro–Munich), and 1 in day 23 (*Vueling* Lisbon–Sevilla). So the selected day is the 22, and the number of canceled flights is 3.

In the second sample input, there is only one flight between airports where *Portforce* operates, but that airline is not operated by *Portforce*. So the maximum number of flights that can be affected is zero, and the first day where that happens is the first day of November.