Qualifying Rounds

Problems

October 1, 2005

A Bancopia

Problem

Gold transports are risky nowadays in Bancopia due to the numerous robberies. Some roads are more dangerous than others. In order to minimize the probability of a robbery on a gold transport between two given cities, the Bancopians want to place police posts on some of their roads, which will make those roads safer.

In this task the input will consist of an overview of the cities in Bancopia and the roads between them, together with the two cities the gold transport is between. For each road, the probability of a robbery is given. With the *robbery probability* of a road between two cities we mean the probability that, given that a gold transport travels over that road, a robbery happens there, on that transport. We assume that when a gold transport travels over a number of roads, the robbery probabilities on these roads are independent.

Finally, the maximum number of police posts is given. On every road at most one police post can be placed. A Bancopian police post exactly halves the robbery probability of that road. The question is how large the probability is of a robbery on a gold transport that takes the safest route between the two given cities when the police posts are placed in such a way that this probability is minimized.

Input

The first line of the input file contains a single number: the number of test cases to follow. Each test case has the following format:

- One line with the five integers n, w, m, a and b, separated by spaces. $2 \leq n \leq 100$ is the number of cities, $1 \leq w \leq 10^4$ the number of roads, $1 \leq m \leq 100$ the maximum number of police posts to be placed, and a and b are the cities between which the transport is to take place. Cities are always designated by an unique number in the set $\{1, \ldots, n\}$.
- w lines, each of which describes one road, with three numbers, separated by spaces: s_1, s_2 and p. s_1 and s_2 are the cities the road connects ($s_1 \neq s_2$, $1 \leq s_1, s_2 \leq n$) and $0 \leq p \leq 1$ is the robbery probability of that road.

No two roads in the input will connect the same pair of cities. A road is always considered bidirectional and the robbery probability does not depend on the direction the road is travelled. Finally, it is guaranteed that there is a route from city a to city b with the roads in the input.

Output

For every test case in the input file, the output should contain a single number, on a single line: the probability that a gold transport that travels along the safest route from a to b is robbed, when at most m police posts are placed in such way that this probability is minimized, rounded to *four* decimals after the decimal point. The rounding should be done as usual: $0.12345\ldots$ is rounded to 0.1235, $0.12344\ldots$ to 0.1234.

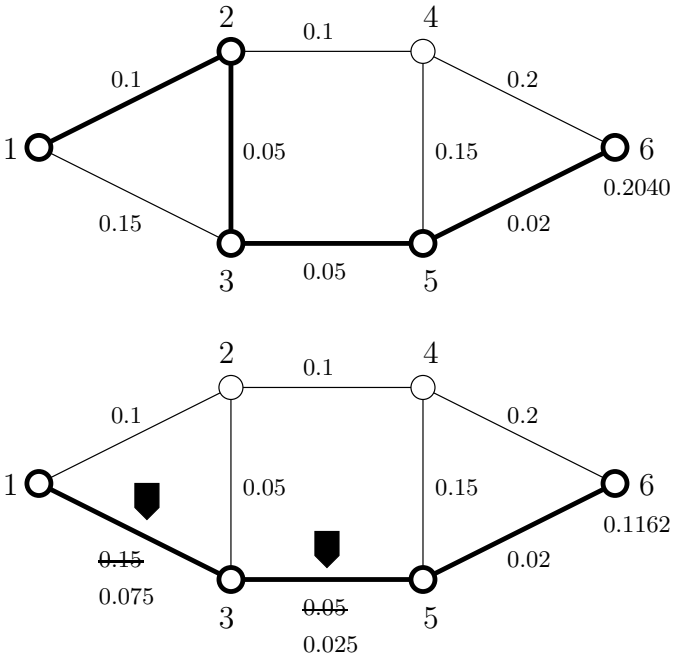


Figure 1: These maps correspond to the example. The first map gives the safest route without police posts; the second one gives the safest route in the solution. The cities are numbered and the robbery probability is shown next to each road. Next to city 6, the destination city, the probability of a robbery on the entire route is given.

Example

Input (see Figure 1)

```
1
6 8 2 1 6
1 2 0.1
1 3 0.15
2 3 0.05
2 4 0.1
3 5 0.05
4 5 0.15
4 6 0.2
5 6 0.02
```

Output

```
0.1162
```

B Flipping Networks

Problem

The Dean of the Unseen University, not unknown to inhabitants of the Discworld or their acquaintances, has decided to modernise his communication by installing a computer network consisting of a number of bidirectionally connected hosts, numbered from 1 to h consecutively. Unfortunately, due to the highly magical nature of the environment, random changes in the network structure occur quite often.

Therefore, it is especially useful to know which hosts can be reached and which cannot. Luckily the changes in the structure can be monitored without further affecting the network and the current network state can therefore be known at any time.

It is agreed upon that any host which can be reached in 10 hops or less from the main host, number 1, is called online. This means that there may be a number of hosts which are reachable from host 1, but are not called online. The Dean wants to know how many such hosts there are.

Input

The first line of the input file contains a single number: the number of test cases to follow. Each test case has the following format:

- One line with one integer, h, the number of hosts ($1 \leq h \leq 3000$);
- One line with one integer, c, the number of initial connections ($1 \leq c \leq 1500$);
- c lines with two integers, p and q, two hosts between which an initial connection exists;
- One line with one integer, l, the number of connection changes ($1 \leq l \leq 1500$);
- l lines with two integers, r and s, two hosts between which a connection (dis)appears (changes from present to absent or vice versa).

Duo to the magical environment, the only condition that applies to p, q, r and s is that they are in the range $1 \ldots h$.

Output

For every test case in the input file, the output should contain a single number, on a single line: the number of hosts reachable from host number 1 which are not considered online.

Example

Input	Output
2 4 4 1 2 2 3 3 4 4 1 4 1 3 1 3 2 3 3 4 12 5 10 11 3 6 9 8 1 4 4 7 8 11 3 5 6 7 10 6 5 5 9 8 2 5 6 2 12	0 1

C Insecurity

Problem

In the history of operating systems security has always been an issue. This has led to many proposals to improve security. One of the areas these proposals aim at is the storage of passwords. Passwords shouldn't be stored readable and one of the ways to store them very safe is by storing them as a code. Nowadays many systems use an md5-hash to store the passwords.

A possible attack on a password storage using hashes is the use of a bunch of commonly used passwords and their encryption. All you need to do is read out the password file and see if anyone has the same hash for a password as one of your known passwords. Although this won't succeed easily on well-maintained servers with technical users, it's a trick that will be quite easy on a general server, where the users don't know much about security.

To solve this problem a proposal is made to encrypt both the usernames and the passwords. If done cleverly, this might indeed work. A small company, however, overestimated the effectivity of this mechanism. They gave the guarantee that, combined with their encryption techniques and a lot of user accounts, one could give out all usernames and passwords without compromising security worse than in the current situation.

Of course, since you see the errors they make, you will try and find evidence that security will be thoroughly compromised. You seek out one of their servers and retrieve the list of usernames, the list of passwords and the passwordfile.

Their encryption works as follows: Let w be the word to be encrypted and $w[0]$ through $w[n-1]$ the characters in this word, represented by 8 bits using normal ASCII encoding. Let $c[i]$ be the encryption of the first $i+1$ characters of w. $c[i]$ is considered to be a bitstring, not a string of characters. The following rules will provide the encryption:

- $c[0] = w[0]$
- $c[i] = (c[i-1] << 4)$ XOR $w[i]$ $(i \geq 1)$

The operation $<<$ is commonly known as a bitwise left-shift. The effect of the operation is shifting all bits to the left. So the bit string 00101011 would become 0010101100 when being left-shifted by 2. For the encryption the bit string can keep growing, so no bits will ever be thrown away. Since zero-bits are important, even if they're the left-most bits and would normally be ignored, the effect of the operator here is adding some zero-bits to the right of the bit string.

The XOR operation is defined as a bitwise XOR, meaning it compares the bits instead of the bit string itself. A bitwise XOR operation produces a 0-bit when the compared bits are equal and a 1-bit otherwise. The following is an example of this: 100011 XOR 0101 = 100110.

The word to be encrypted is a concatenation of the username and the password, so `usernamepassword`.

Input

The first line of the input file contains a single number: the number of test cases to follow. Each test case has the following format:

- One line with the encrypted username and password, which should be cracked. The bitstring will be represented by upper case hexadecimal digits. Every 4 bits are represented by one hexadecimal digit. The bitstring 110010101101 would be represented by CAD.
- One line with m $(1 \leq m \leq 10^5)$, the number of users in the system;
- m lines with every line containing one username;

- m lines with every line containing one password.

The usernames and passwords contain only the letters `a` to `z`, `A` to `Z`, the digits `0` to `9`, and the special characters `_-=+!@#$%{}&*()[]\|/<>,.`. Both username and password have at least 8 and at most 30 characters. When encrypting the ASCII-values of the characters are used.

All usernames are unique. The same holds for the passwords.

Output

For every test case in the input file, the output should contain two strings, each on a separate line: the username and the password that were used to form the concatenation of encrypted username and password given in the input. The input is such that in each case a unique solution exists.

Example

Input

```
2
7237B23A13EC745236
5
amsterdam
teamdelft
eindhoven
enschede
groningen
abcdefgh
ijklmnop
qrstuvwx
yzabcdef
ghijklmn
62652F07224264EB08455
2
skywalker
darthvader
dark_Force
by1XWing
```

Output

```
teamdelft
yzabcdef
darthvader
dark_Force
```

D Mandalas

Problem

Shasita is drawing mandalas. First, she draws some circles, and then she colours the obtained regions.

Her younger sister Rice wants to help her drawing the mandalas. They agree Rice will draw the circles and Shasita will colour them. Shasita does not want to colour a very large number of all small regions and therefore she fixes the number of regions Rice should create by drawing the circles. Rice, on her side, is not very good at counting regions and she asks you to help her.

In this task you have to give the number of regions in a given mandala.

Input

The first line of the input file contains a single number: the number of test cases to follow. Each test case has the following format:

- one line with the integer $0 \leq n \leq 500$, the number of circles in this test case
- n lines, each with the three integers x_i, y_i and r_i, describing the i^{th} circle, which has midpoint (x_i, y_i) and radius r_i, with $|x_i|, |y_i| \leq 1000$ and $0 < r_i \leq 1000$.

Output

For every test case in the input file, the output should contain a single number, on a single line: the number of regions the circles divide the plane $\mathbb{R}^2$ in.

Example

Input	Output
3 1 5 7 10 2 0 0 5 10 0 5 3 0 0 7 10 10 7 0 10 8	2 3 6

E Minotaur

Problem

While investigating an excavation site on the island Crete, the archaeologist Theseus has discovered the entrance to a maze. Afraid of nothing, our brave and daring Theseus entered the maze. After having mapped part of the entire maze, he arrives at a special hall. When he tries to read some text on the wall, suddenly a projection appears in the middle of the hall. It is Minos, Theseus greatest rival and evil enemy, who seems to have found the maze before Theseus did.

Minos' projection tells Theseus that he won't survive his visit to this maze, because as soon as he leaves the hall, a robot, called Minotaur, will be activated that has one and only one mission: track down Theseus and confront him with his fast rotating archaeologist shredder. Minotaur will also be activated in any case after a period of five hours, so Theseus has limited time. "But," says Minos' projection, "to make it a fair game, I will show you a map of the maze with the locations of you, Minotaur and the exit. Also, remember that Minotaur moves twice as fast as you can do." A map shows up in the air, and Theseus sees the following text faintly appearing near Minotaur:

```
function decideDirection()
  if noWall(myPos.westPos())  and victimPos.isWestOf(myPos)  then return(west)
  if noWall(myPos.eastPos())  and victimPos.isEastOf(myPos)  then return(east)
  if noWall(myPos.northPos()) and victimPos.isNorthOf(myPos) then return(north)
  if noWall(myPos.southPos()) and victimPos.isSouthOf(myPos) then return(south)
  return(none)
```

Luckily, Theseus' girlfriend Ariadne has given him a laptop computer as a present just before he left to go to the excavation site that morning. He decides to write a program to help him find an escape route – most preferably within the five hours.

Theseus models his situation as a game in which every time Minotaur has two turns to play, followed by one turn for Theseus. In every turn it is allowed to move either one square west, east, north or south, or hold still for the duration of that turn. Minotaur shreds Theseus as soon as they are on the same square, and Theseus escapes as soon as he enters the Exit square.

In this task *you* must write the program, in the limited time resting of this five hour programming contest...

Input

The first line of the input file contains a single number: the number of test cases to follow. Each test case has the following format:

- One line with two integers, w and h, with $1 \leq w \leq 50$ the number of squares in East-West direction and $1 \leq h \leq 50$ the number of squares in North-South direction.
- h lines with each w characters (excluding the newline character). The meaning of the characters is as follows:
  - `#` indicates a Wall square
  - `.` (a dot) indicates an empty square
  - `T` indicates the initial position of Theseus
  - `M` indicates the initial position of Minotaur
  - `X` indicates the Exit

The characters `T`, `M` and `X` appear all exactly once in each input case. The first line describes the most Northern squares, the last line the most Southern squares. The first character on each line describes the most Western square, the last character the most Eastern one. A wall surrounds the entire maze, but it is *not* included in the input. The exit can be anywhere in the maze, because it is an opening in the roof. It is guaranteed that there is a path of non-Wall squares from Theseus to the Exit.

Output

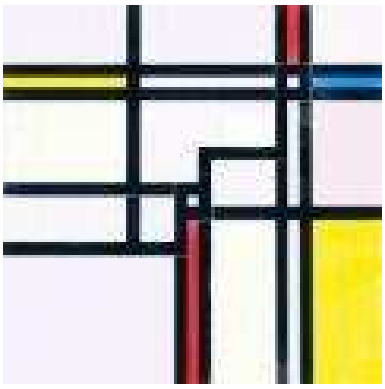
For every test case in the input file, the output should contain a single number, on a single line: the minimum number of turns Theseus needs to escape from the maze, or `0` if Theseus can never reach the Exit safely. The number of turns should include only Theseus' turns, not the ones Minotaur played.

Example

Input	Output
2 10 7 .##.##..#. .T#M#..##X .##.#.##.. ..#.#....# #.#....#.# ..######.. #........# 10 7 .##.##..#. .T#M#..##X .##.#.##.. ..#....#.# #.#.#....# ..######.. #........#	0 20

F Mondriaan

Problem



The great Dutch painter Piet Mondriaan (1872–1944) is considered one of the first modern painters. A major part of his compositions is divided in a couple of rectangular regions. Some of them are filled with a color, while the remaining ones are left white.

Mondriaan's colorings usually obeyed the following rules:

- Every colored region is either red, yellow or blue;
- Two (horizontally or vertically) adjacent regions don't have the same color. White is not considered a color, so two adjacent regions can be both white.

Once a painting is divided in regions, there are still a lot of different ways to fill these with colors. Mondriaan is wondering how many different ways there are for a given division. For the paintings we consider, this number will not exceed 10^6.

Two regions that only touch at a corner point are not considered adjacent.

Input

The first line of the input file contains a single number: the number of test cases to follow. Each test case has the following format:

- One line with one positive number n with $1 \leq n \leq 100$: the number of regions in the painting.
- n lines with four non-negative integers x_1, y_1, x_2 and y_2 with $0 \leq x_1, x_2, y_1, y_2 \leq 10^9$: the coordinates of two opposite vertices of a region. Every region has a non-zero area, regions will not overlap and the union of the regions will form a rectangular painting. (Therefore, the illustration above would be an invalid input case for this problem.)

Output

For every test case in the input file, the output should contain a single number, on a single line: the number of ways the painting can be colored.

Example

Input	Output
2 2 100 110 70 105 100 105 12345 110 4 0 0 1 1 0 1 1 2 1 0 2 1 1 1 2 2	13 121

G Nim/3

Problem

You are staying in the country of Determinisia for a challenging programming contest. The Determinisians are very fond of playing deterministic games, i.e. games of which the result can be known in advance, assuming the players play optimally. They really love to see each time everything is going exactly the way they had foreseen.

The great teacher Oneplusoneistwo has developed the game Nim some thousands of years ago, and this game is still very popular in Determinisia. This game is quite simple: there are three (possibly empty) stacks of matches. By turns, the players must choose a non-empty stack and remove any positive number of matches from it. The first player to make all three stacks empty, wins.

Recently, the scientist Oneplustwoisthree has suggested to make a three player Nim game, called Nim/3. In order to make this game deterministic, each of the players has one of the other two players as a favorite; in case he can't win himself, he will try to let his favorite win. The players must also know each other's favorites.

In order to socialize with the local students, you want to play a few games of Nim/3 with them. But this is only possible when you play the game in an optimal way. Luckily, you are allowed to write a computer program to assist you during game play.

Input

The first line of the input file contains a single number: the number of test cases to follow. Each test case has the following format:

- One line with six integers: s_1 s_2 s_3 f_1 f_2 f_3. Here, s_1, s_2 and s_3 denote the number of matches on the stacks 1, 2 and 3 respectively ($0 \leq s_n \leq 20$, $s_1 + s_2 + s_3 \geq 1$). f_1, f_2 and f_3 indicate the favorite player of player 1, 2 and 3 respectively.

Note that $f_p \in \{1, 2, 3\} \setminus \{p\}$. Currently it is the turn of player 1, followed by player 2 and 3 respectively.

Output

For every test case in the input file, the output should contain two numbers, on a single line, separated by a space: k and n. $k \in \{1, 2, 3\}$ is the stack and $0 < n \leq s_k$ the number of matches to take from that stack by player 1, when he plays an optimal game. There might be multiple optimal choices, but you should give the one with k as small as possible, and next n as small as possible.

Example

Input	Output
3 1 1 5 3 3 2 1 1 5 2 1 1 0 1 4 2 1 1	3 4 1 1 3 1

H Venus Rover

Problem

After the NASA[1] sent their Mars Exploration Rovers *Spirit* and *Opportunity* to Mars, the ASAN[2] decided to send their Venus Exploitation Rover *Greedy* to Venus in order to find out which valuable raw resources can be obtained from Venus. Its mission is to collect some stones from Venus' surface.

Greedy will be transferred to Venus using a rocket that will drop it on the surface together with a large container, then fly seven times around Venus and finally pick up both *Greedy* and the container from the surface using its on-board grabbers.

After its landing, *Greedy* will use its IntelliSensor technology to scan for all interesting stones within half a mile. This will yield a list of stones with accurate estimations of their mass, their value for the Administratika and the time required to pick them up and put them in the container. The container is large enough to contain even all the stones, but the rocket can only lift a limited amount of mass from the surface. Also, the amount of time is limited due to the fact that the rocket will come back after seven rounds around Venus.

It is your task to write the program that will determine which of these stones to pick up and put in the container, if the total value is to be maximized.

Input

The first line of the input file contains a single number: the number of test cases to follow. Each test case has the following format:

- A line with three positive integers: N, T and M. $0 < N \leq 100$ is the number of stones found, $0 < T \leq 100$ the time available before the rocket will come back to pick up *Greedy* and the container and $0 < M \leq 100$ the maximum mass of the stones that the rocket can lift.
- N lines, with each i^{th} line containing the three positive integers t_i, m_i and v_i (all no greater than 10^6), representing respectively the time required for pick-up, the estimated mass and the estimated value for stone i.

Output

For every test case in the input file, the output should contain a single number, on a single line: the maximum total value collectable in the corresponding test case.

[1] National Aeronautics and Space Administration
[2] Administratika Spatika and Aeronautika Nationalika

Example

Input

```
2
1 20 10
2 2 100
5 20 10
6 6 10
10 5 12
5 10 18
12 5 10
3 3 7
```

Output

```
100
19
```