BUBT IUPC

Bangladesh University of Business & Technology: CSE Fiesta 2010

June 27th, 2010 (Original Contest date was June 19th)

Notes to Contestants

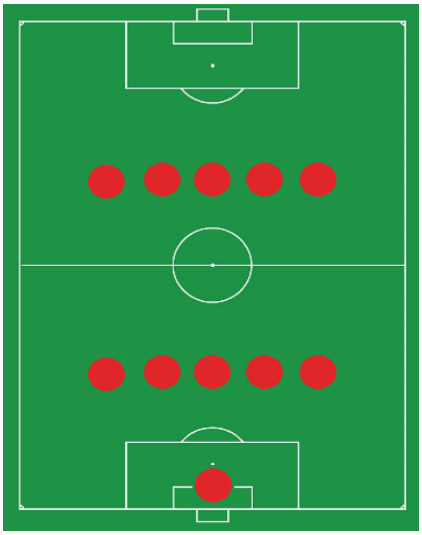
1. There are 9 problems in this set
2. There are a total of 18 pages including this one
3. Contest Duration is 5 hours. In case of unforeseen circumstances, we might increase the duration.
4. Ranklist will be suppressed 45 minutes before the end of the contest.
5. The time limits of each problem will be notified after the contest starts
6. All input should be read from standard input, and all output should be written to standard output
7. Make sure your file names don't contain any space character
8. If you have any queries about the problems, ask for clarifications through PC^2. For other queries ask one of the volunteers.

The Problem Set	**Problem Credits**
A – Argentina B – Bafana Bafana C – Cheerleaders D – Dumb Eucelics' Pyramid Walk E – Ensuring Victory F – Floating-Point Numbers G – Gentle Ping, to the Old King H – Hello, Mafia I – Innovative Procession Management	Shahriar Manzoor Jane Alam Jan Sohel Hafiz Manzurur Rahman Khan Samee Zahur Muntasir Khan Shamim Hafiz Arifuzzaman Arif

"Programming contests are fun, but important fun. Contest problems provide many students with their first serious exposure to algorithm design, and the encouragement to lure them into advanced study of Computer Science. Contest problems provide a primary source of interview questions for high-tech companies like Google and thus have a disproportionate impact on careers of many of their employees. It is no surprise that companies like Google, Microsoft and IBM sponsor student programming competitions in order to get first dibs on the best talent. Success in programming contest is a source of pride for several countries, and influences both their educational policy and the relative ranks of universities and CS departments."

-Steve Skiena

A Argentina



The Argentine football team coach, the great Diego Maradona, is going to try out a new *formation* this year. *Formation* describes how the players are positioned on the pitch. Instead of the conventional 4-4-2 or 4-3-3, he has opted for 5-5. This means there are 5 attackers and 5 defenders.

You have been hired by *Argentina Football Federation* (AFF) to write a code that will help them figure out which players should take the attacking/defensive positions.

Maradona has given you a list containing the names of the 10 players who will take the field. The attacking ability and the defensive ability of each player are also given. Your job is to figure out which 5 players should take the attacking positions and which 5 should take the defensive positions.

The rules that need to be followed to make the decision are:

- The sum of the attacking abilities of the 5 attackers needs to be maximized
- If there is more than one combination, maximize the sum of the defending abilities of the 5 defenders
- If there is still more than one combination, pick the attackers that come lexicographically earliest.

Input

The first line of input contains an integer **T**(**T**<50) that indicates the number of test cases. Each case contains exactly 10 lines. The **i**th line contains the name of the **i**th player followed by the attacking and defending ability of that player respectively. The length of a player's name is at most 20 and consists of lowercase letters only. The attacking/defending abilities are integers in the range [0, 99].

Output

The output of each case contains three lines. The first line is the case number starting from 1. The next line contains the name of the 5 attackers in the format (A_1, A_2, A_3, A_4, A_5) where A_i is the name of an attacker. The next line contains the name of the 5 defenders in the same format. The attackers and defenders names should be printed in lexicographically ascending order. Look at the sample for more details.

Sample Input	Sample Output
1 sameezahur 20 21 sohelh 18 9 jaan 17 86 sidky 16 36 shamim 16 18 shadowcoder 12 9 muntasir 13 4 brokenarrow 16 16 emotionalblind 16 12 tanaeem 20 97	Case 1: (emotionalblind, jaan, sameezahur, sohelh, tanaeem) (brokenarrow, muntasir, shadowcoder, shamim, sidky)

Problemsetter: Sohel Hafiz, Special Thanks: Shamim Hafiz, Jane Alam jan

B Bafana Bafana

Team practice is very important not only for programming contest but also for football. By team practice players can learn cooperating with team mates. For playing as a team improvement of passing skill is very important. Passing is a great way of getting the ball upfield and reduces the risk of giving the ball away.

Carlos Alberto Parreira, the coach of Bafana Bafana, also wants his players to practice passing a lot. That's why, while in the training camp for soccer world cup 2010, every day he asks all of the players who are present in practice to stand in a circle and practice passing. If **N** players are in practice, he gives each of the players a distinct number from 1 to **N**, and asks them to stand sequentially, so that player 2 will stand in right side of player 1 and player 3 will stand in right side of player 2, and so on. As they are in a circle, player 1 will stand right to player N.

The rule of passing practice is, Parreira will give the ball to player **K**, and practice will start. Practice will come to an end after **P** passes. In each pass, a player will give the ball to his partner who is in his immediate right side. After **P** passes, the player who owns the ball at that moment will give the ball back to Parreira.

Parreira wants to be ensured that his players practice according the rule. So he wants a program which will tell him which player will give him the ball back. So after taking the ball from the same person he can be happy that, the players play according to the rules. Otherwise he will ask them to start from beginning.

Input

Input starts with an integer **T** (**T** <= 1000), the number of test cases. Each test case will contain three integers, **N** (2 <= **N** <= 23), **K** (1 <= **K** <= **N**), **P**(1<=**P**<=200).

Output

For each test case, output a single line giving the case number followed by the Bafana player number who will give the ball to Parreira. See sample output for exact format.

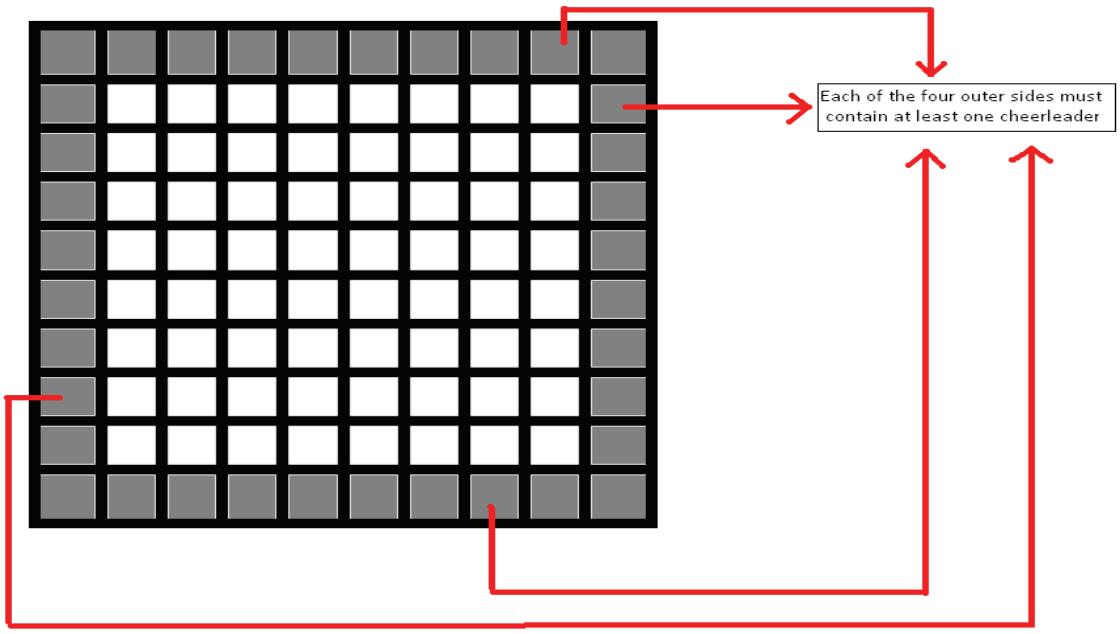
Sample Input	Sample Output
3 5 2 5 6 3 5 4 1 3	Case 1: 2 Case 2: 2 Case 3: 4

Problemsetter: Md. Arifuzzaman Arif, Special Thanks: Muntasir Khan, Sohel Hafiz

C Cheerleaders

In most professional sporting events, cheerleaders play a major role in entertaining the spectators. Their roles are substantial during breaks and prior to start of play. The world cup soccer is no exception. Usually the cheerleaders form a group and perform at the centre of the field. In addition to this group, some of them are placed outside the side line so they are closer to the spectators. The organizers would like to ensure that at least one cheerleader is located on each of the four sides. For this problem, we will model the playing ground as an **M*N** rectangular grid. The constraints for placing cheerleaders are described below:

- There should be at least one cheerleader on each of the four sides. Note that, placing a cheerleader on a corner cell would cover two sides simultaneously.
- There can be at most one cheerleader in a cell.
- All the cheerleaders available must be assigned to a cell. That is, none of them can be left out.



The organizers would like to know, how many ways they can place the cheerleaders while maintaining the above constraints. Two placements are different, if there is at least one cell which contains a cheerleader in one of the placement but not in the other.

Input

The first line of input contains a positive integer **T<=50**, which denotes the number of test cases. **T** lines then follow each describing one test case. Each case consists of three nonnegative integers, **2<=M, N<=20** and **K<=500**. Here **M** is the number of rows and **N** is the number of columns in the grid. **K** denotes the number of cheerleaders that must be assigned to the cells in the grid.

Output

For each case of input, there will be one line of output. It will first contain the case number followed by the number of ways to place the cheerleaders as described earlier. Look at the sample output for exact formatting. Note that, the numbers can be arbitrarily large. Therefore you must output the answers modulo **1000007**.

Sample Input	Sample Output
2 2 2 1 2 3 2	Case 1: 0 Case 2: 2

Problemsetter: Md. Arifuzzaman Arif, Special Thanks: Muntasir Khan, Sohel Hafiz

D Dumb Eucelics' Pyramid Walk

Little ant Eucelics has been given a holy task. He is to walk from anthill Almapores to another anthill Bigopores, all on his own, unguided by any other ant through a flat desert terrain under the scorching heat of the sun. As if that was not enough, there is a huge pyramid structure standing on the ground, which might force him to take a longer route. He has therefore asked you to calculate the shortest possible distance that he must travel in order to reach his goal, walking around or over the pyramid if necessary.

The pyramid has a square base on the ground. For this problem we will use a Cartesian coordinate system with its origin on the ground at the exact center of the base of the pyramid, while axes are drawn such that the corners of the base square are at coordinates (-10,-10), (-10,10), (10,10) and (10,-10).

Input

The input consists of multiple test cases. The first line of the input is an integer **T** (≤150), the number of test cases. This line is followed by **T** more lines, each specifying a single test case. Each of these lines specify five space-separated real numbers are in the format "$\mathbf{A_x}$ $\mathbf{A_y}$ $\mathbf{B_x}$ $\mathbf{B_y}$ **h**", each with at most two digits after the decimal point. The location of Almapores and Bigopores are given by ($\mathbf{A_x}$, $\mathbf{A_y}$) and ($\mathbf{B_x}$, $\mathbf{B_y}$) respectively. The height of the pyramid is given by **h**. All lengths are measured in meters. You may assume that the two anthills at distinct locations, and none of the two anthills lie inside the pyramid. The inputs will be such that $|\mathbf{A_x}|$, $|\mathbf{A_y}|$, $|\mathbf{B_x}|$, $|\mathbf{B_y}| \le 25$, $0 \le \mathbf{h} \le 150$.

Output

You must produce a single line of output for each input test case. Each line should be of the format "Case **c**: **d**", where **c** is the serial number of the case starting from 1, and **d** is the shortest walking distance as asked for in the problem. Your answer will be accepted if absolute error of your output is less than 10^{-6}.

Sample Input	Sample Output
3 -13 0 13 0 50 -13 0 13 0 70 -13 0 13 0 100	Case 1: 40.78474941 Case 2: 40.88061302 Case 3: 40.88061302

Problemsetter: Samee Zahur, Special Thanks: Jane Alam Jan

E Ensuring Victory

Since your team always loses in football, you are now planning to tip the balance in your favor by using a specially designed ball. To this end you construct many balls of varying designs and from several different materials and ask your team to try them out.

As expected, the feedback is rather random and inconsistent, with some players liking some of the balls while others not liking those very same ones. Since that is not helping, you decide on a more objective method of finding out which ones are the best. You conduct a series of experiments and observe the behavior of each of these balls when kicked. After the trials, you have worked out for each ball an exact formula that gives the ball's position t seconds after being kicked.

Using that, you can determine the behavior of the balls in an actual game accurately on paper. You are interested in knowing the position of the ball as well as the speed at which it is travelling at specific moments in time. The position (which is basically its Euclidean distance from a certain point) is found by simply computing the value from the formula. The speed of the ball is defined as the rate of change of distance with time at that moment (i.e. the *derivative* of distance with respect to time). Since there are a lot of different balls, you want to automate this process and get the results before the tournament starts.

Unfortunately, your football team is not any better at computing than they are at football (you must have guessed that already). So it falls again on you to solve this problem.

Input

Input file will consist of a number of test cases (<=50). Each test case consists of two lines. The first line is a simple algebraic expression which gives the distance covered by the ball in terms of **t**. The length of this expression can be at most 200. It will consist of numbers and the operators '*' and '+' and the variable **t**. The numbers will all be fractional numbers of the format “X/Y”, where X is the numerator and Y the denominator. X and Y in this line will never be negative and Y will never be 0. The expression might also contain parentheses ('(' and ')') to force precedence. The usual rules and precedence of arithmetic apply when using the formula to compute the answer. The next line consists of a single fractional number on a line by itself. It is the value of t. This is also of the form “X/Y”, but here X and Y can also be negative. No input line will be greater than 150 characters in length. There will be no extra whitespace or any blank lines in input.

Output

Output will consist of exactly as many lines as there are cases. For each case, print a line of the form "Case #C: A B". C is the case (starting from 1), while A and B are fractional numbers in their simplest form. A is the distance of the ball and B is its speed at time t. Both numbers are also of the form "X/Y". If the value is negative, print it as "-X/Y". Zero should be printed as "0/1".

Note:

1. As these are specially built balls, their behavior will not always be like what you would expect from an ordinary ball. Don't make any assumptions.
2. All calculations can be performed without any intermediate fraction, in their reduced forms, overflowing a 32-bit signed integer in their numerators and denominators.
3. Both the distance and the speed can be negative - these trick footballs can actually move opposite to the direction they were kicked in.
4. A fractional number X/Y is said to be in its most reduced or simplest form when there is no number other than 1 that evenly divides both X and Y. So, 3/1, 2/5 and 11/17 are in their simplest form but 9/3, 5/20 and 10000/500 are not.

Sample Input	Sample Output
((t*t)+2/1) 1/2 ((t+t)*2/1) 1/2 (t*1/2)+(t*1/3) 1/4	Case #1: 9/4 1/1 Case #2: 2/1 4/1 Case #3: 5/24 5/6

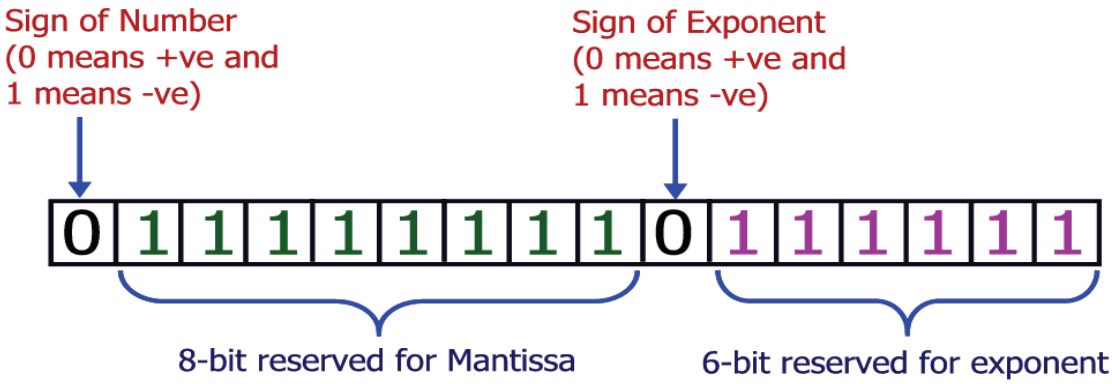
Problemsetter: Muntasir Khan, Special Thanks: Samee Zahur, Jane Alam Jan

F Floating-Point Numbers

Floating-point numbers are represented differently in computers than integers. That is why a 32-bit floating-point number can represent values in the magnitude of 10^{38} while a 32-bit integer can only represent values as high as 2^{32}.

Although there are variations in the ways floating-point numbers are stored in Computers, in this problem we will assume that floating-point numbers are stored in the following way:

Floating-point numbers have two parts mantissa and exponent. M-bits are allotted for mantissa and E bits are allotted for exponent. There is also one bit that denotes the sign of number (If this bit is 0 then the number is positive and if it is 1 then the number is negative) and another bit that denotes the sign of exponent (If this bit is 0 then exponent is positive otherwise negative). The value of mantissa and exponent together make the value of the floating-point number. If the value of mantissa is m then it maintains the constraints $\frac{1}{2} \le m < 1$. The left most digit of mantissa must always be 1 to maintain the constraint $\frac{1}{2} \le m < 1$. So this bit is not stored as it is always 1. So the bits in mantissa actually denote the digits at the right side of decimal point of a binary number (Excluding the digit just to the right of decimal point)
In the figure above we can see a floating-point number where M=8 and E=6. The largest value



this floating-point number can represent is (in binary) $0.111111111_2 \times 2^{111111_2}$. The decimal equivalent to this number is: $0.998046875 \times 2^{63} = 9205357638345293824_{10}$. Given the maximum possible value represented by a certain floating point type, you will have to find how many bits are allotted for mantissa (M) and how many bits are allotted for exponent (E) in that certain type.

Input

The input file contains around 300 line of input. Each line contains a floating-point number F that denotes the maximum value that can be represented by a certain floating-point type. The floating point number is expressed in decimal exponent format. So a number AeB actually denotes the value $A \times 10^B$. A line containing 0e0 terminates input. The value of A will satisfy the constraint 0<A<10 and will have exactly 15 digits after the decimal point.

Output

For each line of input produce one line of output. This line contains the value of M and E. You can assume that each of the inputs (except the last one) has a possible and unique solution. You can also assume that inputs will be such that the value of M and E will follow the constraints: $9 \geq M \geq 0$ and $30 \geq E \geq 1$. Also there is no need to assume that (M+E+2) will be a multiple of 8.

Sample Input	Sample Output
5.699141892149156e76 9.205357638345294e18 0e0	5 8 8 6

Problemsetter: Shahriar Manzoor, Special Thanks: Derek Kisman (Moderator), Shamim Hafiz, Tanaeem M Moosa

G | Gentle ping, to the old King

All of you must have heard the name of *The Lord of the Rings*. As we know that the ring, that was said to be the lord of the rings, was destroyed by *Frodo Baggins* in the mountain of doom. And so, the evil spirit *Sauron* was not able to capture the ring and was finally destroyed. After defeating *Sauron*'s army, *Aragorn* became the king and with his coronation the world moved into the dominion of Men. And they lived happily ever after.

But that's not actually the whole story. Defeating *Sauron* was not the final victory over "evil" as such. Because the evil dark lord and *Sauron*'s master - *Morgoth* can still recover, grow and take shape again.

After the death of *Sauron*, many years have passed. *Aragorn* is old now and his son *Eldarion* is the king. Things were going fine except one day they discovered the fact that *Morgoth* has recovered his powers and is planning to take over the world. The whole middle Earth is again waiting for another war.

Eldarion has made several plans with the wise council to stop *Morgoth*. They have the map, so they are aware of the fact that there are some territories in Middle-Earth. Many two-way roads exist between some pairs of territories. The times to travel all roads are known to them. To go from a territory to another one, they can use one or more roads. And all territories are reachable from any territory.

That's why they have the fear that if *Morgoth*'s army enters to any territory then they can invade any territory they want. To protect a road, huge number of armies has to be placed there.

Eldarion is a man of honor like his father. So, he doesn't want to leave any territory unguarded or weak. But since his armies are little in number compared to *Morgoth*'s, he is at a loss. Cause if he sends armies to protect the roads then they may win, but some territories might go down. But if he sends the armies to protect the territories, then if any territory goes down, more will be in danger because *Morgoth*'s army will forward their march using the unguarded roads and of course they can capture roads and can hamper the communication amongst territories. But if *Eldarion* places armies to protect both roads and territories then both roads and territories will be

weak since the number of armies protecting a territory or a road will be too low to fight against *Morgoth*'s army.

After a long discussion, they have got many ideas, but no idea fascinated *Eldarion*. So, he went to his father, *Aragorn* - the old king. *Aragorn* took a day to think, and the next day he told his son about his idea. The idea is to keep as fewer roads as possible such that the armies can still be sent from any territory to all other territories. Rest of the roads will be destroyed. As they have many trebuchets and catapults, they can destroy roads easily and without heavy effort.

After that they will pick some territories that have the higher chances to be attacked first. These territories will be selected by Aragorn himself from his past experience in battles, and these are called the *prime territories*. All the armies will be assigned to protect these territories only. When a territory is under attack, either prime or non-prime, the informers in this territory will seek help from a *prime territory*. This helping territory will be randomly chosen. After the arrival of the armies, the informers will seek help from another randomly chosen *prime territory*. And they will continue seeking help until armies from all the *prime territories* come. The informers will always choose a new *prime territory* for seeking help. So, the *estimated time* for a territory is the total required time for armies from all *prime territories* to reach this territory. And the *Total estimated time* is the summation of *estimated times* of all territories. You can assume that the time to inform other territories is negligible.

Now, they are planning to use *Aragorn's* idea. So, they want to keep the roads such that the *Total estimated time* is as low as possible. You are one of the head councilors and you know the map fully. Now you have to find the minimum *Total estimated time*. Remember that the whole middle earth is depending on you.

Input

The first line of the input will contain **T (<= 100)**, denoting the number of cases.
Each case starts with a blank line. The next line will contain three integers **n (2<=n <= 16)**, **m** and **k (1 <= k <= n)**, where **n** is the number of territories, **m** is the total number of roads and **k** is the number of *prime territories*. The territories are numbered from **0** to **n – 1**. The next line contains **k** integers separated by spaces. These integers denote the *prime territories*. Each of the next **m** lines will contain three integers **u, v (0 <= u, v < n and u != v)** and **w (1 <= w <= 1000)** denoting that there is a two way road between territory **u** and **v** and the time to cross this road is **w** minutes. You can assume that all the roads are valid, no road is listed more than once and multiple roads between same pair of territories don't exist.

Output

For each case, print the case number and the minimum *Total estimated time*.

Sample Input	Sample Output
2 3 2 1 0 0 1 2 1 2 5 3 3 2 0 2 0 1 2 1 2 5 0 2 50	Case 1: 9 Case 2: 21

Problemsetter: Jane Alam Jan Special Thanks: Md. Arifuzzaman Arif

H Hello, Mafia

Programming contests are now too frequent in Bangladesh. Just 5 years ago, you could participate in only one or two contest per year, but now, there is one in almost every month. Moreover, they are also in different cities around the country. Maybe, someday, you will see contests in every week, or maybe every day.

Due to the schedule of the contests, the contestants are really conscious about optimizing the time spent in traveling. For example, if the contest starts at 9:00AM, they plan to reach contest venue, just before that time, so that, no time is wasted.

Contestants will be using various forms of public transportations like bus, train, taxi, rickshaw to reach the contest spot. Bus and train may be used to transport between cities, taxi and rickshaws may be used to move within cities. You will be given all the routes one can use to move between cities. Since, contestants are optimizing their time spent in the contest, they will always take the fastest route possible. Rickshaws and taxis always use the fastest route, so you don't have to be concerned about them.

As the programming contests have become popular, it also caught interest of the mafias as well. They are planning to sabotage this contest. For that, they will call strike in different cities, and thus, no vehicle enters or leaves the city. You can't even use rickshaws and taxi to move in the city during the strike. So, it won't be possible for any team to travel through the city, and may have to use a longer route. If they can't use their shortest route, they will surely be late, and thus, cannot attend the contest.

Mafia lords have a list of teams. They will call strike in minimum number of cities, so that these teams are unable attend the contest. There may be more than one set of cities they can call strike to sabotage the contest, in those cases, they will probably choose the one that affects minimum number of cities.

You have some how managed to know about this evil plan. Now, given the list of cities, you have to find the how many cities will be affected by it.

Input

Input starts with an integer **T(T≤30)**, the number of test cases.

Each test case starts with two integers **N,M(1≤N≤100000, 0≤M≤500000)** the number of cities and the number of direct routes between cities. Each city is denoted by an integer between 0 and **N**-1. The contest is going to be held in city 0. Each of the next **M** lines each contain three integers, **u, v, t(0≤u, v<N,1≤t≤1000)**, a route between city **u** and **v** that takes time **t**. There will be at most one direct route between two cities.

This is followed by an integer **Q(1≤Q≤1000)**, the number of queries. Next **Q** lines each describe a query. Each query starts with an integer **K(1≤K≤100)**, the number of teams in the list, followed by **K** integers $\mathbf{n_i}$ **(0≤n_i<N)**.

There will be a blank line before each test case.

Output

For each test case, output the case number. For each query, output

```
x possible way(s) to sabotage the contest.
Teams from at least y cities will be late.
```

Where ***x*** is the number of possible ways to sabotage, and ***y*** is the minimum number cities, affected by the strike. If sabotaging is not necessary, output

```
Will not sabotage.
```

For formatting details, see the sample output

Sample Input	Sample Output
3 5 8 0 1 10 0 3 5 0 4 7 1 2 1 1 3 3 2 3 7 2 4 6 3 4 2 2 2 2 4 2 2 3 6 4 0 1 10 1 2 10 2 3 10 3 4 10 1 2 4 2 3 1 0 1 10 1 1 2	Case 1: Query 1: 1 possible way(s) to sabotage the contest. Teams from at least 5 cities will be late. Query 2: 2 possible way(s) to sabotage the contest. Teams from at least 3 cities will be late. Case 2: Query 1: 3 possible way(s) to sabotage the contest. Teams from at least 3 cities will be late. Case 3: Query 1: Will not sabotage.

Problemsetter: Manzurur Rahman Khan, Special Thanks: Jane Alam Jan

I Innovative Procession Management

It's FIFA World Cup 2010. The whole world is waiting for the greatest event on the planet. Since huge number of people will come to South Africa for the world cup; it's quite tough for them to maintain the flow of the people as well as the traffic. That's why they are planning to forbid vehicles to use some roads. But if all the roads are forbidden then people would be in trouble.

The World Cup authority has divided South Africa into **n** junctions and **m** one way roads. Before each match, the authority will consider some junctions as interesting junctions, where people will get together and set out for the stadium in processions. Now each procession will try to use the shortest path towards the stadium. That's why the World Cup authority wants to forbid vehicles in the roads which may be used by the processions.

The idea is as follows. They will first find the shortest path amongst the paths from the interesting junctions to the stadium. They will block all the roads in the path. If there are multiple shortest paths from one or multiple interesting junctions to the stadium, they will block all of them. They will send some vehicles along the paths which will put notice boards to the roads such that no vehicles can go through. After that they will search for more shortest-paths considering the blocked roads and send vehicles for the paths (if any). They will continue blocking the roads until no path is found.

Now your task is to solve the problem for them.

Input

Input starts with an integer **T (<= 100)**, denoting the number of test cases.

Each case starts with a blank line. The next line contains three integers **n (2 <= n <= 100), m (0 <= m <= 10000)**, and **S (1 <= S < n)**. **S** is the total number of interesting junctions.

The next line will contain **S + 1** different integer (each of them lies between **1** and **n**) separated by spaces, where the first **S** integers denote the interesting junctions, and the last one denotes the stadium.

Each of the next **m** lines will contain 3 integers **u v w (1 <= u, v <= n, u != v, 1<=w<=10000)** meaning that there is a road from **u** to **v**, and the length is **w**. If multiple roads are listed for same junctions in either direction, you can assume that the roads are different.

Output

For each case print the case number in a single line. If no path is found then print "**No road to block**". If a shortest path is found then print the path cost **c** as "**The path cost is c**". After that report the roads that are blocked in **u v w** form (as in input). They should be listed in ascending order based on **u;** for same **u**, based on **v**. Report all the shortest path costs and the roads in similar format. Check the samples for more details.

Sample Input	Sample Output
3 2 1 1 1 2 2 1 10 3 5 2 1 3 2 2 1 10 1 2 20 1 2 20 1 2 30 3 2 30 4 4 2 1 2 4 1 4 10 2 4 10 1 3 20 3 4 10	Case 1: No road to block Case 2: The path cost is 20 1 2 20 1 2 20 The path cost is 30 1 2 30 3 2 30 Case 3: The path cost is 10 1 4 10 2 4 10 The path cost is 30 1 3 20 3 4 10

Problemsetter: jane Alam Jan, Special Thanks: Manzurur Rahman Khan