Problem A. Administrative Troubles

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

It is difficult for a spy to do his job without a decent car. The Bureau of Administrative Personnel for Cars (BAPC) spy-car rental company has a large collection of cars that spies can use and handles the resulting administration. Using cars obviously costs money: they require gasoline to operate and repairs are needed quite often, because spies tend to cause accidents a little more often than other drivers.

At the end of the year, all spies need to be billed for the usage of cars in the past year. Last week, there was a major crash in the billing system, rendering it unusable. All that could be recovered was a list of all available types of cars and a log of events for the past year. Using this information, the spy-car rental company wants to obtain a list of the costs for car usage for every spy on record. This list can then be used to send out the bills manually.

Every type of car is registered with a catalog price, the cost to pick up the car and the cost of driving that car per kilometer. The event list contains three types of events: pick-ups, returns and accidents. When a spy picks up a car, he or she must pay the pick-up cost for that car. Once the car is returned, the number of kilometers driven in the car is recorded and the spy must pay for these kilometers. If an accident occurred when the spy was using the car, repairs need to be paid for. Every accident is rated with a severity as a percentage. To repair the car, this percentage of the catalog price is billed to the spy who caused the accident. If any billed cost is fractional, it is rounded up before being added to the bill.

The list of all available types of cars is complete. However, because of the crash, some events in the recovered event log might be missing. The spy-car rental company does not want to present spies with an inconsistent bill, so you should detect inconsistencies in the log entries for each spy. The following conditions hold for a consistent event log:

- A spy will pick up a car before returning it.
- A spy will always return a car they picked up.
- A spy can use at most one car at a time.
- Accidents can only happen when a spy is using a car.

Input

On the first line one positive number: the number of test cases, at most 100. After that per test case:

- one line with two space-separated integers n and m ($0 \le n \le 500$ and $0 \le m \le 10^4$): the number of types of cars, and the number of events, respectively.
- n lines with a string N and three integers p, q and k ($1 \le p \le 10^5$, $1 \le q \le 1000$, $1 \le k \le 100$), all separated by a space: for each type of car, its unique name, its catalog price, its pick-up cost, and its cost per kilometer driven, respectively.
- m lines starting with one integer t ($0 \le t \le 10^5$), a string S and one character e, all separated by a space: the time of the event, the name of the involved spy, and the type of event, followed by:
  — if e is equal to '`p`' (pick-up): a string C: the name of the type of car picked up.
  — if e is equal to '`r`' (return): an integer d ($0 \le d \le 1000$): the distance covered in the car last picked up by spy S, in kilometers.
  — if e is equal to '`a`' (accident): an integer s ($0 \le s \le 100$): the severity of the accident, in percents.

All names of cars and spies consist of at least 1 and at most 40 lowercase letters. There will be at most 500 unique spy names in each test case. The events are given in chronological order.

Output

Print one line for every spy referenced in any of the events, containing a string and one integer, separated by a space: the name of the spy and his total car cost. If the event log for the spy is inconsistent, the total cost should be replaced by the string "`INCONSISTENT`".

The lines should be sorted alphabetically by the name of the spy.

Example

standard input	standard output
1 2 8 bmw 5000 150 10 jaguar 7000 200 25 10 mallory p bmw 15 jb p jaguar 20 jb r 500 35 badluckbrian a 100 50 mallory a 10 55 silva p jaguar 60 mallory r 100 110 silva a 30	badluckbrian INCONSISTENT jb 12700 mallory 1650 silva INCONSISTENT

Problem B. Bribe

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

After having done a lot of spying and infiltrating a criminal network, you are now ready to try and dismantle it. This, however, requires the cooperation of a certain number of the henchmen. This in turn requires money in order to bribe them, but due to budget cuts, you only have a limited amount of money.

Fortunately, you are an excellent judge of character, so for each of the henchmen you are considering to bribe, you know what amount of money they will ask for. Furthermore, you know the probability that they will then successfully convert, as opposed to taking the money and making a run for it. There is no particular rush, so after each attempted conversion you can establish whether it was successful or not, before you move on to someone else. Of course, if it was not successful, then you cannot try to bribe this henchman a second time.

Given all this information on the henchmen, the amount of money that you have at your disposal, and the number of henchmen you need to convert, can you work out the probability that this operation will be a success?

Input

On the first line one positive number: the number of test cases, at most 100. After that per test case:

- one line with three space-separated integers n, c and m ($1 \leq n, c \leq 16$ and $1 \leq m \leq 1000$): the number of henchmen that are susceptible to bribe, the number you need to convert, and the amount of money that you have, respectively.
- n lines with two space-separated integers b and p ($0 \leq b \leq 1000$ and $0 \leq p \leq 100$): the amount of money you need to bribe each henchman, and the probability (as a percentage) that he will be successfully converted, respectively.

Output

Print one line with a single floating point number: the probability that you will succeed in converting c henchmen, if you take an optimal approach. This number should be accurate up to 10^{-6} relative or absolute precision.

Example

standard input	standard output
2 4 3 1000 300 40 300 50 300 60 300 70 4 2 1000 100 80 700 50 400 20 500 20	0.21 0.408

Problem C. Code

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

A terrorist organisation known as the New World Ensemble for Rebellious Coders (NWERC) is a menace to our society. Fortunately, we have found a way to intercept their commu- nications without them knowing it. There is a problem however, since their messages are encrypted.

What we do know from intelligence is that their messages consist of lowercase letters only, and that the encryption method is the Basic Alphabet Permutation Code (BAPC). In this code, every same letter is replaced with another (lowercase) letter from the alphabet (or possibly the same letter). Obviously, two different letters in the original message are still different after encryption, so that there is a unique decryption. So for example, "`hello`" might be encrypted as "`ifmmp`" or "`holle`", but not as "`cnoiz`" or "`bgrrb`".

Still, this leaves a lot of possibilities, and our efforts to break the code have been in vain so far. Fortunately, we had a stroke of luck: through an informant, we were able to get hold of a decrypted message D. We do not have the corresponding encrypted message, but we are fairly certain that it must be somewhere in a list of intercepted encrypted messages we have.

Your task is to write a program which, given the original message D and a list of encrypted messages, works out which encrypted letter belongs to which decrypted letter (assuming that one of the encrypted messages corresponds to the original message), for as much as is possible.

This should then be used to decrypt a freshly intercepted encoded message X, as much as possible. Specifically, each letter of X for which it can be determined with absolute certainty what it represents — either directly or deductively — should be decrypted. All other letters should be replaced by a question mark.

Note that, even when multiple messages can correspond to D, it may still be possible to decrypt certain letters (see for example the third test case in the sample input).

Input

On the first line one positive number: the number of test cases, at most 100. After that per test case:

- one line with a single integer n ($1 \leq n \leq 100$) the number of encrypted messages.
- n lines with a single string M_i: the encrypted messages.
- one line with a single string D: the decrypted message, assumed to correspond to one of the encrypted messages.
- one line with a single string X: the message to be decoded.

All strings consist of at least 1 and at most 1000 lowercase letters.

Output

Print one line with a single string Y . For each letter in X, the corresponding letter in Y should be the decrypted counterpart, or a '`?`' if it is unknown. If there is no encrypted message M_i that could possibly be the encrypted version of D, then the string should be "`IMPOSSIBLE`" instead.

Example

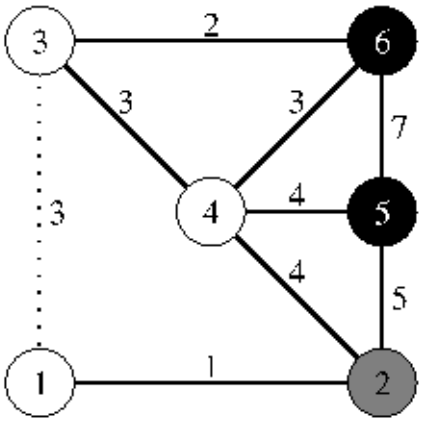
standard input	standard output
3 3 mtahovcjqxels irajsbkticlur gnubipwdgkryf aboringsample rbunyfka 2 ejotydins xchmrwbcxg decrypted dmvenw 2 abccdeb afccdgf message abcdefg	problem? IMPOSSIBLE m?sa???

Problem D. Destination Unknown

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

You are agent B100. A pair of prominently dressed circus artists is traveling over the roads of the city and your mission is to find out where they are headed. All we know is that they started at point s and that they are heading for one of several possible destinations. They are in quite a hurry, though, so we are sure they will not take a detour to their destination.

Alas, prominently dressed as they may be, the duo is nowhere to be seen. Fortunately, you have an exceptional sense of smell. More specifically: your nose will never let you down. You can actually smell they have traveled along the road between intersections g and h. Where is the elusive duo headed? Or are we still not sure?



A visual representation of the second sample. The duo is travelling from the gray circle to one of the two black circles, and you smelled them on the dashed line, so they could be heading to 6.

Input

On the first line one positive number: the number of test cases, at most 100. After that per test case:

- One line with three space-separated integers n, m and t ($2 \leq n \leq 2000$, $1 \leq m \leq 50000$ and $1 \leq t \leq 100$): the number of intersections in the city, the number of individual roads between those intersections, and the number of possible destinations respectively.
- One line with three space-seperated integers s, g and h ($1 \leq s, g, h \leq n$): the intersection the duo started from and the two intersections between which the duo has traveled, with $g \neq h$.
- m lines with three space-separated integers a, b and d ($1 \leq a < b \leq n$ and $1 \leq d \leq 1000$), indicating that there is a bidirectional road between intersections a and b of length d.
- t lines with one integer x ($1 \leq x \leq n$): the possible destinations. All possible destinations are distinct and they are all different from s.

There is at most one road between a pair of intersections. One of the m lines describes the road between g and h. This road is guaranteed to be on a shortest path to at least one of the possible destinations.

Output

Print one line with one or more space-separated integers, indicating the destinations that the duo can still be headed for, in increasing order.

Example

standard input	standard output
2 5 4 2 1 2 3 1 2 6 2 3 2 3 4 4 3 5 3 5 4 6 9 2 2 3 1 1 2 1 1 3 3 2 4 4 2 5 5 3 4 3 3 6 2 4 5 4 4 6 3 5 6 7 5 6	4 5 6

Problem E. Encoded Coordinates

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	512 mebibytes

You have been monitoring a terrorist cell that is planning a big attack somewhere close. The terrorists have been sending encoded coordinates to each other, and you suspect that the attack will happen at one of the locations described by these coordinates. You have intercepted the relevant communications for a series of locations you want to investigate.

Every location consists of a separate x and y coordinate, each being a non-negative integer smaller than some prime number P. Both coordinates are encoded separately using the same process. A source has given you information on the process of decoding a single coordinate. The input of this process consists of five values: A, B, C, K and N.

Central to the decoding process is a collection of three functions, defined in terms of each other and the input value K:

$$F(n+1) = G(n) + H(n)$$

$$G(n+1) = K \cdot F(n) + H(n-1)$$

$$H(n+1) = F(n) + K \cdot G(n)$$

The values A, B and C are used to initialize these functions as follows: $F(1) = A$, $G(1) = B$, $H(1) = C$. The decoded coordinate is the value of $F(N)$ mod P.

As you might have noticed, there is some crucial information missing: you need the value of $H(0)$ to calculate $G(2)$. While you do not have this value, you do know that the x and y coordinate use the same value for $H(0)$. Luckily, you have managed to already obtain the x coordinates for the locations. Your colleagues have suggested that this does not give you enough information to calculate the y coordinates, but you want to prove them wrong.

Input

On the first line one positive number: the number of test cases, at most 100. After that per test case:

- one line with the prime number P ($2 \leq P \leq 19997$).
- one line with five space-separated integers A_x , B_x , C_x , K_x and N_x ($0 \leq A_x, B_x, C_x, K_x < P$ and $1 \leq N_x \leq 10^9$): the parameters for the x coordinate.
- one line with five space-separated integers A_y, B_y, C_y, K_y and N_y ($0 \leq A_y, B_y, C_y, K_y < P$ and $1 \leq N_y \leq 10^9$): the parameters for the y coordinate.
- one line with one integer x ($0 \leq x < P$): the x coordinate.

Output

Print one line with one integer: the calculated y coordinate. If your colleagues are right, and there is not exactly one valid y coordinate, print "`UNKNOWN`" on a single line instead.

Example

standard input	standard output
2 11 2 3 5 7 4 4 6 8 9 7 5 7 1 6 2 5 1 3 4 2 5 3 1	7 UNKNOWN

Problem F. Flying Safety

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Due to budget cuts, even spies have to use commercial airlines nowadays to travel between cities in the world. Although this mode of travel can be very convenient for a spy, it also raises a problem: the spy has to trust the pilot to make sure he is not in danger during the flight. And even worse, sometimes there is no direct flight between some pairs of cities, so that the spy has to take multiple flights to get to the desired location, and thus has to trust multiple pilots! To limit the trust issues you are asked for help. Given the flight schedule, figure out the smallest set of pilots that need to be trusted, such that the spy can safely travel between all cities.

Input

On the first line one positive number: the number of test cases, at most 100. After that per test case:

- one line with two space-separated integers n ($2 \leq n \leq 1000$) and m ($1 \leq m \leq 10^4$): the number of cities and the number of pilots, respectively.
- m lines with two space-separated integers a and b ($1 \leq a, b \leq n$, $a \neq b$): a pilot flying his plane back and forth between city a and b.

It is possible to go from any city to any other city using one or more flights. In other words: the graph is connected.

Output

Print one line with an integer: the minimum number of pilots that need to be trusted such that it is possible to travel between each pair of cities.

Example

standard input	standard output
2 3 3 1 2 2 3 1 3 5 4 2 1 2 3 4 3 4 5	2 4

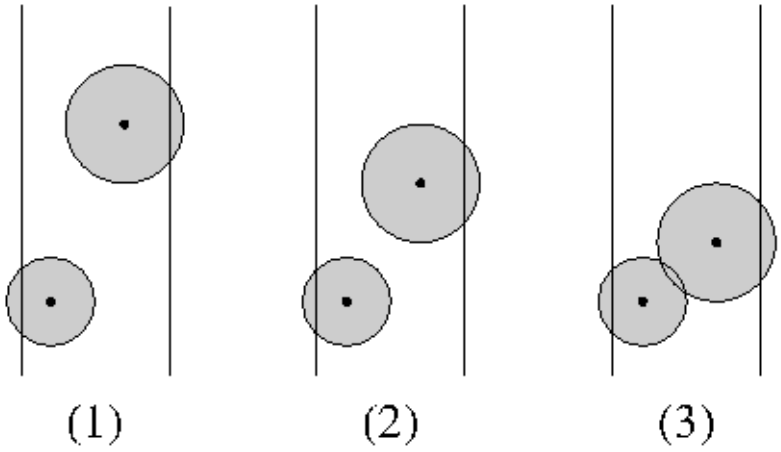
Problem G. Getting Through

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	512 mebibytes

A long and straight corridor is fitted with a number of sensors. Each sensor has a certain range within which it can detect objects or persons. If a part of an object is within the sensors range, an alarm will go off. Otherwise, nothing will happen.

Ethan needs to traverse this corridor in order to do some spy stuff at the other end. The question is, can he pass through the corridor without being detected? Can he fit so easily that he can bring some equipment along, or does he have to wear some tight clothing? Or can he perhaps sent a robot through instead?

We model the corridor as being two-dimensional (we ignore the height), bounded by two straight lines. Each sensor is located inside the corridor or on a wall. Their scopes are well in between the two ends of the corridor. We model the person or robot going through as a circle. Given the layout, what is the maximum radius this circle can have so that it is possible to negotiate the corridor without being detected?



A visual representation of the samples.

Input

On the first line one positive number: the number of test cases, at most 100. After that per test case:

- one line with a single integer w ($1 \leq w \leq 10^5$): the width of the corridor. The two walls are given by the lines $x = 0$ and $x = w$.
- one line with a single integer n ($0 \leq n \leq 1000$): the number of sensors in the corridor.
- n lines with three space-separated integers x, y and r ($0 \leq x \leq w$, $-10^5 \leq y \leq 10^5$ and $1 \leq r \leq 10^5$): the location and the range of each sensor, respectively.

The two ends of the corridor are far beyond the scope of all the sensors.

Output

Print one line with one floating point number: the radius of the largest circular object (or person) that could pass through the corridor without being detected, assuming the object can (be) move(d) with absolute precision. If nothing could possibly get through, the output should be zero. The number should be accurate up to 10^{-6} absolute precision.

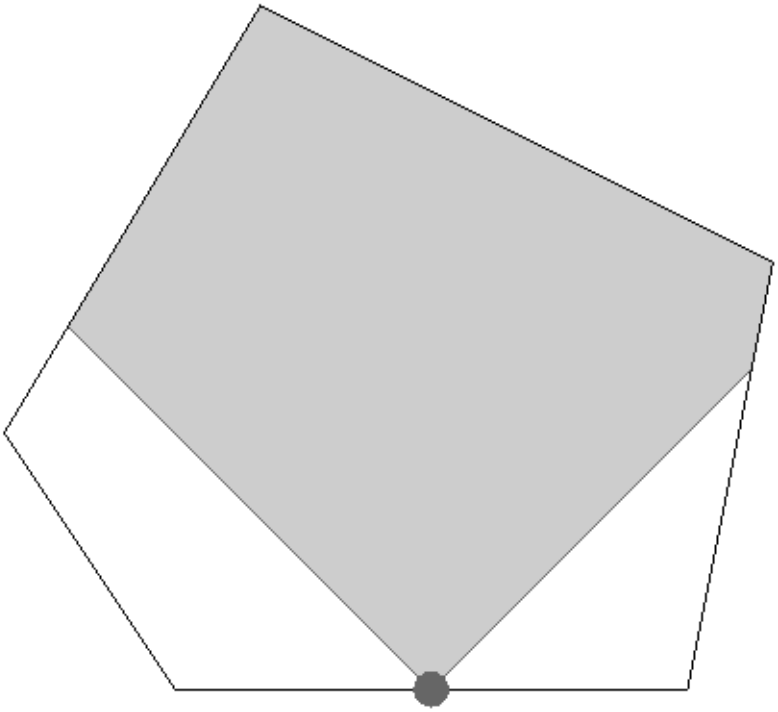
Example

standard input	standard output
3 10 2 2 0 3 7 12 4 10 2 2 0 3 7 8 4 10 2 2 0 3 7 4 4	1.5 1.216991 0

Problem H. Hidden Camera

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

John wants to put a hidden camera in a room. For this problem, we ignore the vertical dimension and treat the room as a two-dimensional object. The room has the shape of a convex polygon. The camera is placed on a wall, halfway between two corners. The camera has a limited view: the borders of the view are given by the two lines that intersect the wall at a 45 degree angle. John wants to know how much of the room is visible to the camera. Can you help him?



The room as described in the first sample. The camera (the dot) can view the shaded region. The limits are given by lines that make a 45 degree angle with the base wall. The area of the shaded region is 71.25% of the total area of the room.

Input

On the first line one positive number: the number of test cases, at most 100. After that per test case:

- one line with a single integer n ($3 \le n \le 1000$): the number of corners of the room.
- n lines with two space-separated integers x and y ($-10^4 \le x, y \le 10^4$): the coordinates of the corners.

The corners are given in counterclockwise order. All angles are strictly between 0 and 180 degrees. The camera is placed exactly halfway between the first two corners in the input.

Output

Print one line with one floating point number: the ratio of the area that the camera can see and the total area of the room. This number should be accurate up to 10^{-6} relative or absolute precision.

Example

standard input	standard output
2 5 -3 0 3 0 4 5 -2 8 -5 3 4 0 2 2 0 3 1 1 3	0.7125 0.5

Problem I. Incognito

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Spies use attributes to disguise themselves to make sure that they are not recognized. For example, when putting on sunglasses, a spy suddenly looks completely different and cannot be recognized anymore. Every combination of attributes gives a different appearance, but not all combinations are possible.

For example, a hat and a turban are both headgear and cannot be used at the same time. Given the list of available attributes, compute how many distinct disguises can be made.

Input

On the first line one positive number: the number of test cases, at most 100. After that per test case:

- one line with an integer n ($0 \leq n \leq 30$): the number of available attributes.
- n lines with two space-separated strings: the name and the category of the attribute.

All strings consist of at least 1 and at most 20 lowercase letters. Within a test case all names are distinct.

Output

Print one line with an integer: the number of possible distinct disguises that can be made with the given attributes, such that at most one attribute from each category is used.

Example

standard input	standard output
2 3 hat headgear sunglasses eyewear turban headgear 3 mask face sunglasses face makeup face	5 3

Problem J. Jailbreak

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

John is on a mission to get two people out of prison. This particular prison is a one-story building. He has managed to get hold of a detailed floor plan, indicating all the walls and doors. He also knows the locations of the two people he needs to set free. The prison guards are not the problem — he has planned a diversion that should leave the building practically void.

The doors are his main concern. All doors are normally opened remotely from a control room, but John can open them by other means. Once he has managed to open a door, it remains open. However, opening a door takes time, which he does not have much of, since his diversion will only work for so long. He therefore wants to minimize the number of doors he needs to open. Can you help him plan the optimal route to get to the two prisoners?

Input

On the first line one positive number: the number of test cases, at most 100. After that per test case:

- one line with two space-separated integers h and w ($2 \leq h, w \leq 100$): the width and height of the map.
- h lines with w characters describing the prison building: '`.`' is an empty space, '`*`' is an impenetrable wall, '`#`' is a door, '`$`' is one of the two people to be liberated.

John can freely move around the outside of the building. There are exactly two people on the map. For each person, a path from the outside to that person is guaranteed to exist.

Output

Print one line with a single integer: the minimum number of doors John needs to open in order to get to both prisoners.

Example

standard input	standard output
3 5 9 ****#**** *..#.#..* ****.**** *$#.#.#$* ********* 5 11 *#********** *$*...*...* *$*.*.*.*.* *...*...*.* *********.* 9 9 *#**#**#* *#**#**#* *#**#**#* *#**.**#* *#*#.#*#* *$##*##$* *#******#* *.#.#.#.* *********	4 0 9