Problem A. Score Sequence

Description

Teacher Liu teaches two classes in a middle school. One class is a good class called "experimental class", the other is a ordinary class. After every exam, Teacher Liu would like to do some comparison between the scores of two classes. This time, after the latest exam, Teacher Liu wants to sort the scores of two classes by descending order respectively to get two score sequences. Then he wants to find some relationship between two sorted score sequences. For example, he wants to find out the longest common sub sequence of these two sorted score sequences (the sub sequence must be consecutive). Would you help him?

Input

There are multiple test cases.

The first line of the input is an integer T(T <=15), indicating the number of test cases.

In each test case:

The first line contains two integers n1 and n2(0<n1,n2<=30), meaning the students number of two classes.

The second line is the score sequence of one class. It contains n1 integers which are all scores.

The third line is the score sequence of the other class. It contains n2 scores.

All scores are from 0 to 100.

Output

For each test case, you should sort those two score sequences by descending order and print the longest sub sequence of the sorted sequences. For the same scores in a sequence, you should just keep one before sorting. If there are more than one longest sub sequences, print the one with the highest score.

After that, you need to sort the sub sequence according to the units digit by ascending order. If two scores have the same units digit, the small one goes first. Print the re-sorted result in a line.

If there is no common sub sequence in the first step, just print "NONE".

Sample Input

```
4
4 4
10 10 30 20
50 50 30 20
8 8
```

```
10 38 27 55 44 66 75 66
38 27 77 44 55 66 66 98
7 8
50 40 30 20 10 5 25
50 40 30 20 10 5 24 23
4 4
10 20 30 40
50 60 70 80
```

Sample Output

```
30 20
20 30
66 55 44 38 27
44 55 66 27 38
50 40 30
30 40 50
NONE
```

Problem B. Power Station

Description

The massive tsunami that struck the coastal city has washed away many of inhabitants and facilities there. After the tsunami, the power supply facilities of the coastal city are completely destroyed. People are in panic in the dark night. Doubts remain over whether the communities will be able to rebuild the city. To calm people down, the heads of the city are planning to rebuild the city to start with the recovery of the power supply facilities.

The coastal city consists of *n* communities which are numbered from 1 to n. To save the electric cables, *n-1* cables has been used to connect these communities together, so that each pair of communities is able to transfer electronic energy mutually.

The heads of the city decide to set a power station in one of the communities. There is thermal energy loss along the cables. Each cable has a resistance of *R* ohm. The total thermal energy loss is the sum of I^2R_i. Here R_i is the total residence along the path between the ith community and the power station, and *I* is a constant. They are troubling their head on the issue of where to set the power station to make the total thermal energy loss minimized.

Input

There are multiple test cases.

The first line contains one integer indicating the number of test cases.

For each test case, the first line contains three positive integers *n*, *I* and R, indicating the number of communities, the above mentioned constant and the residence of each cable. ($3 \le n \le 50000$, $1 \le I \le 10$, $1 \le R \le 50$)

The next *n-1* lines, each describe a cable connection by two integers *X*, *Y*, which indicates that between community *X* and community *Y*, there is a cable.

Output

For each test case, please output two lines.

The first line is the minimum total thermal energy loss.

The second line is all the optional communities in ascending order.

You are requested to leave a blank line after each test case.

Sample Input

```
2
5 1 1
3 2
1 2
5 2
4 3
```

```
6 1 2
1 2
2 3
3 4
2 6
3 5
```

Sample Output

```
5
2

14
2 3
```

Problem C. Sightseeing

Description

CC and MM arrive at a beautiful city for sightseeing. They have found a map of the city on the internet to help them find some places to have meals. They like buffet restaurants (self-service restaurants) and there are n such restaurants and m roads. All restaurants are numbered from 1 to n. Each road connects two different restaurants. They know the price of every restaurant. They go by taxi and they know the taxi fee of each road.

Now they have Q plans. In each plan, they want to start from a given restaurant, pass none or some restaurants and stop at another given restaurant. They will have a meal at one of those restaurants. CC does not want to lose face, so he will definitely choose the most expensive one among the restaurants which they will pass (including the starting one and the stopping one). But CC also wants to save money, so he want you to help him figure out the minimum cost path for each plan.

Input

There are multiple test cases in the input.

For each test case, the first line contains two integers, n, m(1<=n<=1000, 1<=m<=20000),meaning that there are n restaurants and m roads.

The second line contains n integers indicating the price of n restaurant. All integers are smaller than 2×10^9.

The next m lines, each contains three integers: x, y and z(1<=x, y <=n, 1<=z<=2×10^9), meaning that there is a road between x and y, and the taxi fee of this road is z.

Then a single line containing an integer Q follows, meaning that there are Q plans (1<=Q<=20000).

The next Q lines, each contains two integers: s and t (1<=s, t <= n) indicating the starting restaurant and stopping restaurant of each plan.

The input ends with n = 0 and m = 0.

Output

For each plan, print the minimum cost in a line. If there is no path from the starting restaurant to the stopping restaurant, just print -1 instead.

Print a blank line after each test case.

Sample Input

```
6 7
1 2 3 4 5 6
1 2 1
2 3 2
3 4 3
```

```
4 5 4
1 5 5
2 5 2
1 4 3
5
1 4
2 3
1 5
3 5
1 6
2 1
10 20
1 2 5
1
1 2
0 0
```

Sample Output

```
7
5
8
9
-1

25
```

Problem D. Garden

Description

There are n flowerpots in Daniel's garden. These flowerpots are placed in n positions, and these n positions are numbered from 1 to n. Each flower is assigned an aesthetic value. The aesthetic values vary during different time of a day and different seasons of a year. Naughty Daniel is a happy and hardworking gardener who takes pleasure in exchanging the position of the flowerpots.

Friends of Daniel are great fans of his miniature gardens. Before they visit Daniel's home, they will take their old-fashioned cameras which are unable to adjust the focus so that it can give a shot of exactly **k consecutive flowerpots**. Daniel hopes his friends enjoy themselves, but he doesn't want his friend to see all of his flowers due to some secret reasons, so he guides his friends to the best place to catch the most beautiful view in the interval [x, y], that is to say, to **maximize the sum of the aesthetics values of the k flowerpots** taken in one camera shot when they are only allow to see the flowerpots between position x to position y.

There are m operations or queries are given in form of (p, x, y), here p = 0, 1 or 2. The meanings of different value of p are shown below.

1. *p* = 0 set the aesthetic value of the pot in position *x* as *y*. (1 <= *x* <= *n*; -100 <= *y* <= 100)
2. *p* = 1 exchange the pot in position *x* and the pot in position *y*. (1 <= *x*, *y* <= *n*; *x* might equal to *y*)
3. *p* = 2 print the maximum sum of aesthetics values of one camera shot in interval [*x*, *y*]. (1 <= *x* <= *y* <= *n*; We guarantee that *y*-*x*+1>=*k*)

Input

There are multiple test cases.

The first line of the input file contains only one integer indicates the number of test cases.

For each test case, the first line contains three integers: n, m, k (1 <= k <= n <= 200,000; 0 <= m <= 200,000).

The second line contains n integers indicates the initial aesthetic values of flowers from position 1 to position n. Some flowers are sick, so their aesthetic values are negative integers. The aesthetic values range from -100 to 100. (**Notice**: The number of position is assigned 1 to n from left to right.)

In the next m lines, each line contains a triple (p, x, y). The meaning of triples is mentioned above.

Output

For each query with p = 2, print the maximum sum of the aesthetics values in one shot in interval [x, y].

Sample Input

```
1
5 7 3
-1 2 -4 6 1
2 1 5
2 1 3
1 2 1
2 1 5
2 1 4
0 2 4
2 1 5
```

Sample Output

```
4
-3
3
1
6
```

Problem E. Chinese Repeating Crossbow

Description

In Chinese history, Zhuge Liang, prime minister of Shu in three kingdoms period, is regarded as the embodiment of wisdom. When he was dying he passed a book to his apprentice Jiang Wei privately. This book recorded the introduction and specification of a most powerful weapon at that time, called Chinese repeating crossbow or Zhuge repeating crossbow (Figure 1). This weapon can shoot many arrows in a very short time and makes enemies very hard to defense it.

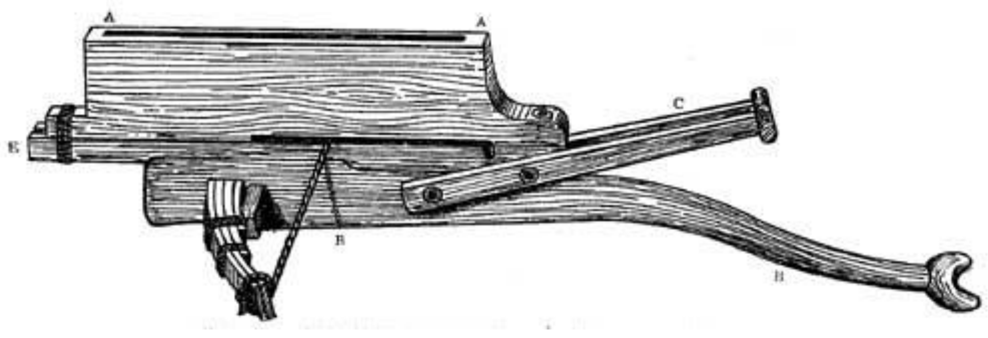


Figure 1 Chinese Repeating Crossbow

Later on, Jiang Wei built a repeating crossbow according to the book and he wanted to know exactly about its power. He came to the center of a test field. The test field was ragged with several extreme high straw walls. Jiang Wei wanted to choose a direction, shot through as many straw walls as he can by his new weapon.

The problem given to you is quite simple: assuming that the repeating crossbow can shot through any numbers of straw walls, please help Jiang Wei to choose a certain direction that he can shot through maximum number of walls in one shot. The walls can be considered as line segments, and if an arrow touches the end points of a wall, it's also called "shoot through". The straw walls can intersect or overlap with each other, and Jiang Wei may possibly stand extremely close to one of the straw wall.

Input

The first line of the input contains an integer T (T<=10) representing the number of test cases.

In each test case, the first line contains an integer N (0<N<=1500) representing the number of straw walls on the field. Each of the following N lines contains four integer numbers x_1, y_1, x_2, y_2， which describes the end point (x_1, y_1) and (x_2, y_2) of a straw wall. The last line of the test case contains two integer (x, y) representing the position of Jiang Wei. All the coordinates are integer numbers and in the range of [-10000, 10000].

Output

For each test case, output a single integer representing the maximum number of straw walls can be shot through in one shot.

Sample Input

```
2
3
-1 -1 1 -1
-1 1 1 1
-1 2 1 2
0 0
3
-1 -1 1 -1
-1 0 0 1
1 0 0 1
0 0
```

Sample Output

```
2
2
```

Problem F. Chess

Description

There is a special chess game. In the game, two people play with go pieces ("x" and "o") on a 4×4 go board. The "x" piece plays first, and players alternate in placing their pieces on an empty cell. The winner is the first player to get an unbroken row of four pieces horizontally, vertically, or diagonally, as shown below:

x			
	x	o	
	o	x	o
			x

Figure 1

x			
	x	x	
o	o	o	o
			x

Figure 2

x	o	o	x
o	x	o	x
x	o	o	o
o	x	x	x

Figure 3

In Figure 1, player with piece 'x' win the game.
In Figure 2, player with piece 'o' win the game.
In Figure 3, there is a tie.

XXX often plays that game with his girlfriend. Sometimes he wants to win the game in order to prove his cleverness. Sometimes he wants to lose the game to let his girlfriend happy. Sometimes he wants to end in a tie (which means that no one wins when there is no empty cell on the board), so that the relationship between his girlfriend and him can be improved.

Here is the question: given the status of middle stage of the board, judge whether XXX can get the result he expected whatever how his girlfriend places her pieces later on.

Input

The input begins with a line containing an integer, indicating the number of test cases. There are no more than 100 test cases.

For each case, the first line is either "WIN", "LOSE" or "TIE", indicating the expectation result of XXX. The next four lines give a 4*4 matrix indicating the status of middle stage of the board. The matrix is formed with the character "x" ("x" piece), "o" ("o" piece) and"." (empty cell). There are 6 to 10 pieces on the board. The number of "x" pieces is equals to or one more than the number of "o" pieces. It is XXX's turn to place the piece. It is promised that no one has won the game in the given board.

Output

For each test case, output a string "YES" or "NO" in a line to tell XXX if he can achieve his expectation.

Sample Input

```
3
LOSE
.o.x
.o.o
x...
.x..
WIN
.o..
xxoo
x..o
x..x
TIE
...o
xo.x
x.xo
o...
```

Sample Output

```
NO
NO
YES
```

Problem G. Necklace

Description

XXX is preparing a necklace which consists of m (4<=m<=100000) diamonds for his girlfriend. He has n (4<=n<=25) kinds of diamonds. Each kind of diamonds has a specific attribution. There are four attributions described as “A”, “B”, “C” and “D”. To make the necklace more beautiful, he wants to follow some rules:

If the attribution of the i-th kind of diamonds is “A”, the number of the i-th kind of diamonds in the necklace must be an odd number.

If the attribution of the i-th kind of diamonds is “B”, the number of the i-th kind of diamonds in the necklace must be an even number.

If the attribution of the i-th kind of diamonds is “C”, the number of the i-th kind of diamonds in the necklace must at least be one.

If the attribution of the i-th kind of diamonds is “D”, the number of the i-th kind of diamonds in the necklace has no constraints.

Following the above rules, XXX want to choose some diamonds and connect them into a circular necklace.The repetitions that are produced by rotation around the center of the circular necklace are neglected. How many different forms of the necklace are there?

NOTE: Because of some ineffable reason, the repetitions that are produced by reflection to the axis of symmetry of the necklace should not be neglected.

Input

There are no more than 50 test cases. The input ends by 0 0.

For each case, the first line begins with two integers --- the above mentioned n and m. On the second line, there is a string with n characters indicating the attribution of each kind of diamonds.

Output

For each case, output one integer in one line, representing the remainder of the number of different forms of the necklace when divided by 10,007.

Sample Input

```
4 4
ABCD
4 4
DDDD
0 0
```

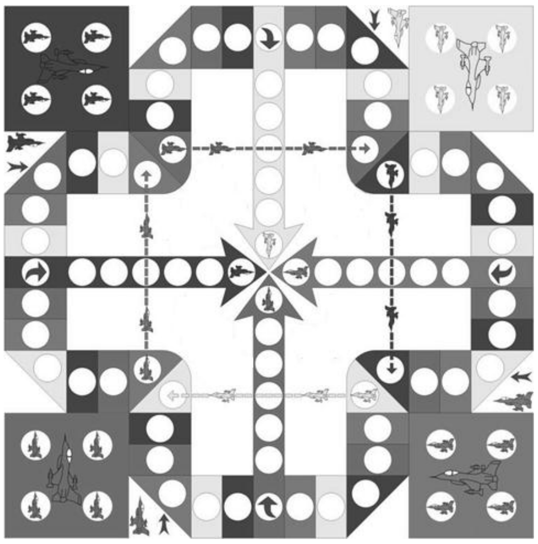
Sample Output

```
11
70
```

Problem H. Aeroplane Chess

Description

Aeroplane Chess is really a popular Chinese board game, and almost everyone had a lot of fun playing it during their childhood. Now we are going back to the past and pick up the memory pieces of those days.



The picture above shows the game board of Aeroplane Chess. There are four "hangers" in each corner, a "track" of 52 spaces, as well as four "home zones" leading from the track to the "destination" at the center.

The goal of the each player is: get all the four pieces of his/her own from the "hanger" into the "destination". Once a player achieves the goal, he/she will be the winner and the game will stop immediately. Each player takes a turn by rolling the standard 6-sided dice, and moves **exactly one piece** of his/her own according to the dice value (unless no pieces can be moved legally).

To make it fair, before the game starts, each player will roll the dice, and the player who gets the highest dice value will be the first one to play. (If there is a tie, the players with the highest dice value will roll again, and then we choose the one with the new highest dice value among them as the first player; Repeat it until there is no tie any more.) After that, the players will take turns in clockwise order.

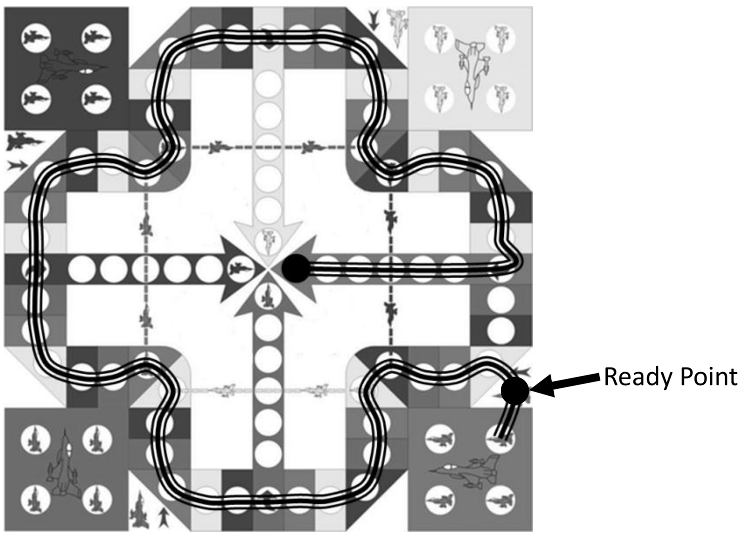
While the game board is symmetric and each player is in the same situation, we just explain the moving rule by taking the player starting at the right-bottom corner as the example.

Basic Moving Rules:

1. At the beginning, all the four pieces are staying in the "hanger".
2. A piece can leave the "hanger" and get ready at the "Ready Point" **only when the dice value is 5 or 6, otherwise it cannot be moved legally**.

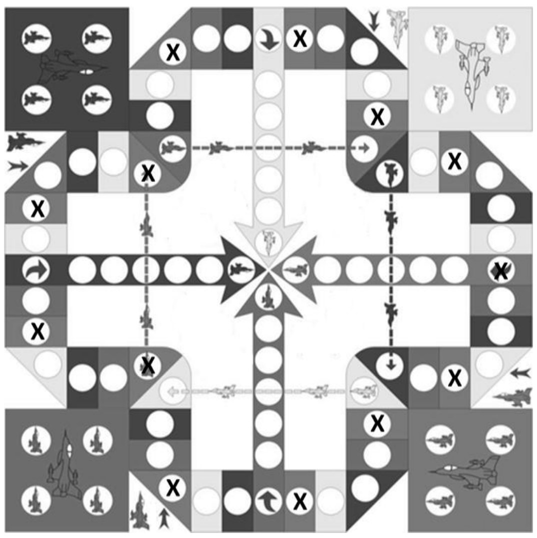
3. Then the piece travels around the “track” in **clockwise** direction, and enters its “home zone”.
4. A piece will move forward by dice value steps and **land on** the target space basically, both on the “track” and in the “home zone”.
5. If a piece arrives at the “destination”, it will never be moved any more.

The picture below shows the whole route of a piece:

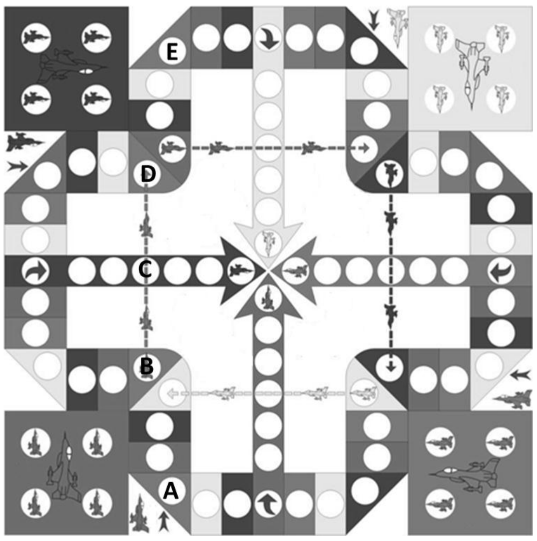


Advanced Rules:

1. If a player gets a roll of 6, he/she will receive a bonus turn immediately.
2. When a piece **lands on** a space of its own color on the “track” (the interaction point of the “track” and the “home zone” is **NOT** included), it **jumps to** the next space of its own color. For it is hard to recognize the different colors while the picture is printed in black and white, the next picture shows the space of the same color as the right-bottom player by marking an X on each of them:



3. There are two types of "shortcut" (the only dotted line of which both the two ends are of the same color as the player) will be used during the game:
   a) A piece **lands on** Space A, and it **jumps to** Space B according to the previous rule. Then it will take the "shortcut" by **jumping to** Space C, Space D orderly.
   b) A piece **lands on** Space B, then it will take the "shortcut" first, **jumping to** Space C, Space D orderly, and then **jump to** Space E.



4. When a piece **lands on** or **jumps to** a space, it will "shoot down" other plays' pieces on the same space, and the "damaged" pieces will return to their own "hanger".
5. A piece should be moved into the "destination" by an exactly dice value, otherwise it will be moved backward by the exceeding steps.

   To simplify this problem, we assume that all players will take the following strategy (**in descending order of the priority**):

1. Move the piece which can shoot down the maximum number of enemies' pieces.
2. Move a piece out of the "hanger" is better than other actions.
3. The piece which can move **forward** further will be considered first. (Hint: the number of steps moved forward may be zero or even negative.)
4. Finally, move the piece which is the most close to the "destination".

Now, there are four players A, B, C, D (A starts at the right-bottom corner, B at the left-bottom, C at the left-top, D at the right-top) playing Aeroplane Chess under the rules explained above, and each player use his/her own dice. Your task is to record some important events during the game.

As is known to all, the dice values are random numbers. However, here we use a simple linear congruential generator to generate them: each dice has a "Feature Number" R; every time we roll this dice, the new "Feature Number" of it will be R' = (R * 123 + 59) mod 65536, and the dice values in this roll will be R' mod 6 + 1.

Input

The input begins with a line containing an integer T (1 <= T <= 20), the number of test cases.

Each test case contains only one line of four integers R_{0A}, R_{0B}, R_{0C}, R_{0D} (0 <= R_{0A}, R_{0B}, R_{0C}, R_{0D} < 65536) indicating the initial "Feature Number" of each corresponding player's dice.

Output

For each test case, you need to print the records of all important events in chronological order. The important events are (in order to avoid confusion, here we **use a '~' to represent a space**):

1. Player X shoots down a piece of Player Y, records like:
   [Player~X]~Shoot~down~a~Player~Y's~plane.
2. Player X moves a piece to the destination, records like:
   [Player~X]~A~plane~arrives~at~destination.
3. Player X wins the game, records like:
   ~Player~A~is~the~winner!~

Sample Input

```
1
1 2 3 4
```

Sample Output

```
[Player C] Shoot down a Player A's plane.
[Player A] Shoot down a Player B's plane.
[Player D] Shoot down a Player A's plane.
[Player D] Shoot down a Player A's plane.
[Player B] Shoot down a Player C's plane.
[Player A] Shoot down a Player C's plane.
[Player C] Shoot down a Player B's plane.
[Player D] Shoot down a Player C's plane.
[Player A] Shoot down a Player D's plane.
[Player D] Shoot down a Player A's plane.
[Player B] Shoot down a Player A's plane.
[Player D] Shoot down a Player B's plane.
[Player C] Shoot down a Player B's plane.
[Player B] Shoot down a Player C's plane.
[Player D] Shoot down a Player B's plane.
[Player A] Shoot down a Player D's plane.
[Player C] Shoot down a Player D's plane.
[Player B] Shoot down a Player A's plane.
[Player A] Shoot down a Player C's plane.
[Player A] Shoot down a Player B's plane.
```

```
[Player A] Shoot down a Player B's plane.
[Player D] A plane arrives at destination.
[Player B] Shoot down a Player C's plane.
[Player D] Shoot down a Player B's plane.
[Player D] Shoot down a Player B's plane.
[Player D] Shoot down a Player C's plane.
[Player A] Shoot down a Player B's plane.
[Player A] Shoot down a Player C's plane.
[Player A] Shoot down a Player D's plane.
[Player A] A plane arrives at destination.
[Player C] Shoot down a Player D's plane.
[Player A] Shoot down a Player D's plane.
[Player B] Shoot down a Player C's plane.
[Player A] A plane arrives at destination.
[Player B] Shoot down a Player C's plane.
[Player B] Shoot down a Player C's plane.
[Player C] A plane arrives at destination.
[Player A] Shoot down a Player D's plane.
[Player B] A plane arrives at destination.
[Player A] A plane arrives at destination.
[Player C] Shoot down a Player D's plane.
[Player D] Shoot down a Player B's plane.
[Player A] A plane arrives at destination.
** Player A is the winner! **
```

Problem I. Hrinity

Description

In the Christian religion, the Trinity is the union of the Father, the Son, and the Holy Spirit in one God. Recently in a far-far-away country, a new word "Hrinity" was created and became very popular. "Hrinity" means "The son is the father, and the father is the son." But the word "Hrinity" has nothing to do with God or any religion. It's about a writer and his son.

When the son was in high school, he failed all exams of all courses. As a none famous writer, the son's worrying father carried out a bold plan: he wrote a long novel and declared that it's his 16 year old son's work. An idiot kid can write a long novel? That made a press interested and the press published the novel. Since then, the son got famous and rich, and his father has been keep writing novels and articles on his son's name.

But the father doesn't trust his son because his son is a punk. Afraid of being treated badly by his son when becoming old, the father embedded "text finger prints" in all the novels and articles. If the son treats him badly, the father will stand out and declare that his son is a fake writer. The father can point out the "text finger prints" in those works to prove that he actually is the author. A text finger print is a delicate sentence which if you add、delete or change the punctuations in it, it's meaning will become totally different. In all his works, the father write many text finger prints whose meaning can be changed into something like "my father wrote this article"、"I am a fake writer", etc. In case of forgetting where the finger prints are, the father needs a computer program to find out how many finger prints are there in a given article. He offers $20,000,000 to buy the program. Do you want this money?

To simplify the problem, we assume that the articles are all written in capital English letters and without blanks.

Input

There are multiple test cases. The first line in the input is an integer T (T<= 15) indicating the number of test cases.

For each test case:

The first line is a integer n(0 < n <= 2500) indicating the number of text finger prints.

Then n lines follows, each represents a text finger print. It’s guaranteed that these n strings are all different. A finger print at least consists of one letter.

The last line of a test case is the article.

The text finger prints and the article may be described in a compressed format. A compressed string consists of capital letters and “compressors”. A “compressor” is in the following format:

[qx]

q is a number(0 < q <= 5,000,000)and x is a capital letter. It means q consecutive letter xs in the original uncompressed string. For example, [6K] means ‘KKKKKK’ in the original string. So, if a compressed string is like:

AB[2D]E[7K]G

It actually is ABDDEKKKKKKKG after decompressed to original format.

The length of the article is at least 1 and at most 5,100,000, no matter in the compressed format or after it is decompressed to original format. The length of a text finger print is no more than 1,100, no matter compressed or original.

Output

For each test case, print an integer K in a line meaning that the article includes K text finger prints. PLEASE NOTE: If finger print s1 is a sub string of finger print s2 and s2 is included in the article, then s1 doesn't count (s1 can be regarded as doesn't exists). Multiple appearances of a finger print in the article are just counted as one.

Sample Input

```
4
2
AB
DCB
DACB
3
A
AB
ABC
DABC
2
[2A]
[3A]B
[5A]B[4A]B
3
AB
CD
EF
ABCDEF
```

Sample Output

```
0
1
1
3
```

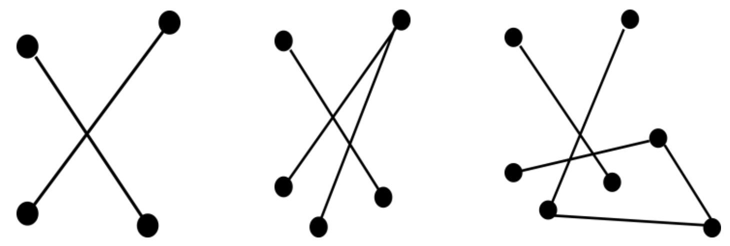
Problem J. World of Goo

Description

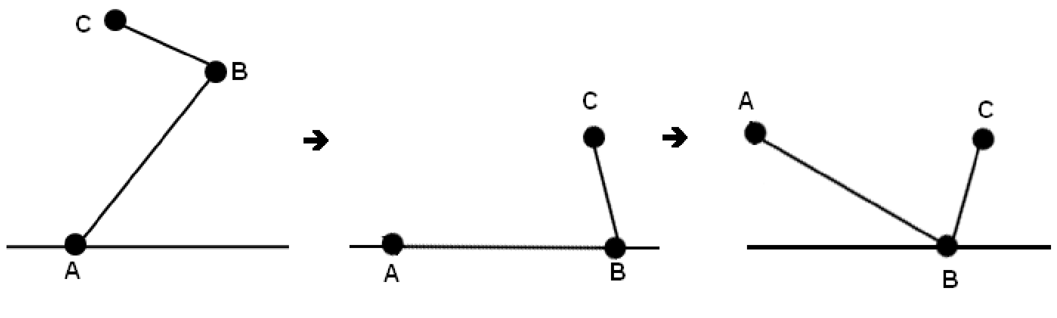
"World of Goo is a physics-based puzzle game by 2D Boy, an independent game developer consisting of Kyle Gabler and Ron Carmel, both former Electronic Arts employees, released for Microsoft Windows, Mac OS X, Linux, iOS, Android and WiiWare. It was nominated for the Seumas McNally grand prize, Design Innovation Award, and Technical Excellence at the Independent Games Festival, and has gone on to win several other gaming awards since its release."

---Wikipedia

Now, Zero, the problem setter has a goo-construction which is made up from "spin goo". You can make a goo-construction by connecting some spin goos with some bars and joining those bars. Bars and goos are all on a same plain. For example, three goo-constructions are shown below:



A spin goo always self-rotates anticlockwise. When a spin goo of a goo-construction reaches the ground, because the friction between the goo and the ground is huge, the goo will sticks on the ground and its rotation will make the whole construction spin clockwise. For example:



At first, the goo A firmly sticks on the plain and the goo B is hang in the air. Goo A starts spinning anticlockwise which results in the whole construction spins clockwise. When the goo B reaches the surface of ground, it will do the same thing as goo A did. As Goo B rotates, goo A no long sticks on the ground.   Next time goo C will stick on the ground and make goo A and Goo B leave the ground.

As time goes on, the goo-construction approaches the exit point(a certain given point) step-by-step. But the goo-construction is such a tremendous burden to spin goos that the construction will slow down spin speed of the goos. If a goo P with an original speed $V_{ori}(P)$ is connected with n goos by n bars, and the weight of bar i is $W_i(1<=i<=n)$, we can get a formula to calculate the actual speed of goo P:

$$V_{act}(P) = \frac{V_{ori}(P)}{\prod W_i}$$

There is no guarantee that the goo-construction is supposed to be on the ground. If the all of the construction is hang in midair, it will fall down with an accelerated speed g=9.8. When falling, it doesn't rotate. y = 0 is the tround.

Now, Zero wonders how much time(in seconds) does a goo-construction need to touch the exit point. Please find the answer to this question, with your excellent programming skills.

Input

There are multiple test cases. Each test case shows a goo-construction and a exit point.

The first line of each case consists of two positive numbers, N and M (0 < N <= 50, 0<M<=200). N represents the number of goos in the construction. M means that there are M bars connecting N goos.All goos are numbered from 0 to N -1.

In the next N lines, each line includes 3 real numbers, x_i, y_i and z_i. (x_i,y_i) is the coordinate of the ith goo(y_i>=0) . Z_i is the original speed of th ith Goo, meaning spinning Z_i rounds per second. It's guaranteed that y_i and z_i are non-negative.

In the following M lines, each line has 3 numbers, u_i ,v_i and w_i , meaning that there is a bar connecting goo u_i and goo v_i , and the bar's weight is w_i . w_i is a real number which is greater than or equal to one.

The last line of a test case has 2 real numbers representing the coordinate of the exit point.

The input ends by N = 0 and M = 0。

Output

For each case, if the goo-construction can touch the exit point, print how many seconds does it need in whole progress(round the result to 2 digits after decimal point). Otherwise print “Bad goo!”(without quotes).

Sample Input

```
4 4
0 0 1
0 1 1
1 0 1
1 1 1
```

```
0 1 1
1 3 1
3 2 1
2 1 1
2 1
0 0
```

Sample Output

```
0.25
```