Problem A. Alice and Bob

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 mebibytes

Alice and Bob are very smart guys and they like to play all kinds of games in their spare time. The most amazing thing is that they always find the best strategy, and that's why they feel bored again and again. They just invented a new game, as they usually did.

The rule of the new game is quite simple. At the beginning of the game, they write down N random positive integers, then they take turns (Alice first) to either:

1. Decrease a number by one.
2. Erase any two numbers and write down their sum.

Whenever a number is decreased to 0, it will be erased automatically. The game ends when all numbers are finally erased, and the one who cannot play in his(her) turn loses the game.

Here's the problem: Who will win the game if both use the best strategy? Find it out quickly, before they get bored of the game again!

Input

The first line contains an integer $T(1 \leqslant T \leqslant 4000)$, indicating the number of test cases. Each test case contains several lines.

The first line contains an integer $N(1 \leqslant N \leqslant 50)$.

The next line contains N positive integers $A_1 \ldots A_N(1 \leqslant A_i \leqslant 1000)$, represents the numbers they write down at the beginning of the game.

Output

For each test case in the input, print one line: "`Case #X: Y`", where X is the test case number (starting with 1) and Y is either "`Alice`" or "`Bob`".

Examples

standard input	standard output
3 3 1 1 2 2 3 4 3 2 3 5	Case #1: Alice Case #2: Bob Case #3: Bob

Problem B. Break the Chocolate

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 mebibytes

Benjamin is going to host a party for his big promotion coming up.

Every party needs candies, chocolates and beer, and of course Benjamin has prepared some of those. But as everyone likes to party, many more people showed up than he expected. The good news is that candies are enough. And for the beer, he only needs to buy some extra cups. The only problem is the chocolate.

As Benjamin is only a 'small court officer' with poor salary even after his promotion, he can not afford to buy extra chocolate. So he decides to break the chocolate cubes into smaller pieces so that everyone can have some.

He have two methods to break the chocolate. He can pick one piece of chocolate and break it into two pieces with bare hand, or put some pieces of chocolate together on the table and cut them with a knife at one time. You can assume that the knife is long enough to cut as many pieces of chocolate as he want.

The party is coming really soon and breaking the chocolate is not an easy job. He wants to know what is the minimum number of steps to break the chocolate into unit-size pieces (cubes of size $1 \times 1 \times 1$). He is not sure whether he can find a knife or not, so he wants to know the answer for both situations.

Input

The first line contains an integer $T(1 \leqslant T \leqslant 10^4)$, indicating the number of test cases.

Each test case contains one line with three integers $N, M, K(1 \leqslant N, M, K \leqslant 2000)$, meaning the chocolate is a cube of size $N \times M \times K$.

Output

For each test case in the input, print one line: "`Case #X: A B`", where X is the test case number (starting with 1) , A and B are the minimum numbers of steps to break the chocolate into $N \times M \times K$ unit-size pieces with bare hands and knife respectively.

Examples

standard input	standard output
2 1 1 3 2 2 2	Case #1: 2 2 Case #2: 7 3

Problem C. Construct the Great Wall

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	256 mebibytes

A defensive wall is a fortification used to protect a city or settlement from potential aggressors. From ancient to modern times, they were used to enclose settlements.

Generally, these are referred to as city walls or town walls. Even though, our ancestors decided to build a Great Wall to protect the northern borders of the Chinese Empire against invasion by various nomadic groups.

The map is given as an rectangle area of size $N \times M$. Each grid is an empty area, a city or an enemy. The Great Wall is a simple polygon build alone the edge of the grids, enclosing all the cities and keeping all the enemies out.

The Great Wall is not easy to build, so we should make the Great Wall as short as possible. Now it is your job to calculate the length of the shortest Great Wall so that it can protect all the cities from the enemies.

Input

The first line contains an integer $T(1 \leqslant T \leqslant 50)$, indicating the number of test cases.

Each test case contains several lines.

The first line contains two integer $H, W(1 \leqslant H, W \leqslant 8)$, indicating the number of rows and columns of the map.

The following H lines contains W chars, indicating the map. '`o`' represents a city, '`.`' represents a empty area and '`x`' represents an enemy.

You can assume that there will be at least one city on the map.

Output

For each test case in the input, print one line: "`Case #X: Y`", where X is the test case number (starting with 1) and Y is the length of the shortest Great Wall (-1 if impossible).

Examples

standard input	standard output
3 3 3 .o. .x. o.o 4 4 .... .ox. .xo. .... 5 5 .ooo. .x... ..xoo x.xoo .ox.x	Case #1: 14 Case #2: -1 Case #3: 28

Note

A simple polygon is a closed polygonal chain of line segments in the plane which do not have points in common other than the common vertices of pairs of consecutive segments.

The solution for the first test case in sample is shown in Figure 1 below. There is no solution for the second test case because no matter how you build the Great Wall, it will always intersects with itself (see Figure 2).

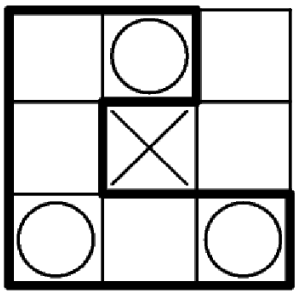

Figure.1

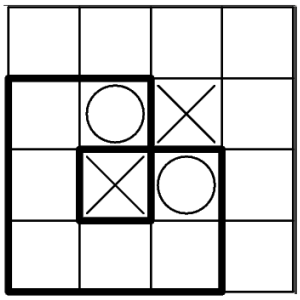

Figure.2

Problem D. Disney's FastPass

Input file:	`standard input`
Output file:	`standard output`
Time limit:	9 seconds
Memory limit:	256 mebibytes

Disney's FastPass is a virtual queuing system created by the Walt Disney Company. First introduced in 1999 (thugh the idea of a ride reservation system was first introduced in world fairs), FastPass allows guests to avoid long lines at the attractions on which the system is installed, freeing them to enjoy other attractions during their wait. The service is available at no additional charge to all park guests. (From Wikipedia)

You are given the map of the whole park, and there are some attractions that you are interested in. How to visit all the interested attractions within the shortest time?

Input

The first line contains an integer $T(1 \leqslant T \leqslant 25)$, indicating the number of test cases. Each test case contains several lines. The first line contains three integers $N, M, K(1 \leqslant N \leqslant 50, 0 \leqslant M \leqslant N(N-1)/2, 0 \leqslant K \leqslant 8)$, indicating the number of locations(starting with 1, and 1 is the only gate of the park where the trip must be started and ended), the number of roads and the number of interested attractions. The following M lines each contains three integers $A, B, D(1 \leqslant A, B \leqslant N, 0 \leqslant D \leqslant 10^4)$ which means it takes D minutes to travel between location A and location B. The following K lines each contains several integers $P_i, T_i, FT_i, N_i, F_{i,1}, F_{i,2} \ldots F_{i,N_i-1}, F_{i,N_i}$, $(1 \leqslant P_i, N_i, F_{i,j} \leqslant N, 0 \leqslant FT_i \leqslant T_i \leqslant 10^4)$, which means the i^{th} interested attraction is placed at location P_i and there are N_i locations $F_{i,1}, F_{i,2} \ldots F_{i,N_i}$ where you can get the FastPass for the i^{th} attraction. If you come to the i^{th} attraction with its FastPass, you need to wait for only FT_i minutes, otherwise you need to wait for T_i minutes. You can assume that all the locations are connected and there is at most one road between any two locations. Note that there might be several attractions at one location.

Output

For each test case in the input, print one line: "`Case #X: Y`", where X is the test case number (starting with 1) and Y is the minimum time of the trip.

Examples

standard input	standard output
2 4 5 2 1 2 8 2 3 4 3 4 19 4 1 6 2 4 7 2 25 18 1 3 4 12 6 1 3 4 6 2 1 2 5 1 4 4 3 1 1 3 2 1 3 4 1 2 4 10 2 8 3 1 4 4 8 3 1 2	Case #1: 53 Case #2: 14

Problem E. Eliminate the Conflict

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 mebibytes

Conflicts are everywhere in the world, from the young to the elderly, from families to countries. Conflicts cause quarrels, fights or even wars. How wonderful the world will be if all conflicts can be eliminated.

Edward contributes his lifetime to invent a 'Conflict Resolution Terminal' and he has finally succeeded. This magic item has the ability to eliminate all the conflicts. It works like this:

If any two people have conflict, they should simply put their hands into the 'Conflict Resolution Terminal' (which is simply a plastic tube). Then they play 'Rock, Paper and Scissors' in it. After they have decided what they will play, the tube should be opened and no one will have the chance to change. Finally, the winner have the right to rule and the loser should obey it. Conflict Eliminated!

But the game is not that fair, because people may be following some patterns when they play, and if the pattern is founded by others, the others will win definitely.

Alice and Bob always have conflicts with each other so they use the 'Conflict Resolution Terminal' a lot. Sadly for Bob, Alice found his pattern and can predict how Bob plays precisely. She is very kind that doesn't want to take advantage of that. So she tells Bob about it and they come up with a new way of eliminate the conflict:

They will play the 'Rock, Paper and Scissors' for N round. Bob will set up some restricts on Alice. But the restrict can only be in the form of "you must play the same (or different) on the i^{th} and j^{th} rounds". If Alice loses in any round or break any of the rules she loses, otherwise she wins.

Will Alice have a chance to win?

Input

The first line contains an integer $T(1 \leqslant T \leqslant 50)$, indicating the number of test cases. Each test case contains several lines. The first line contains two integers $N, M(1 \leqslant N \leqslant 10000, 1 \leqslant M \leqslant 10000)$, representing how many round they will play and how many restricts are there for Alice. The next line contains N integers $B_1, B_2, \ldots, B_N$, where B_i represents what item Bob will play in the i^{th} round. 1 represents Rock, 2 represents Paper, 3 represents Scissors. The following M lines each contains three integers $A, B, K(1 \leqslant A, B \leqslant N, K = 0 \text{ or } 1)$ represent a restrict for Alice. If K equals 0, Alice must play the same on A^{th} and B^{th}round. If K equals 1, she must play different items on A^{th} and B^{th}round.

Output

For each test case in the input, print one line: "`Case #X: Y`", where X is the test case number (starting with 1) and Y is "`yes`" or "`no`" represents whether Alice has a chance to win.

Examples

standard input	standard output
2 3 3 1 1 1 1 2 1 1 3 1 2 3 1 5 5 1 2 3 2 1 1 2 1 1 3 1 1 4 1 1 5 1 2 3 0	Case #1: no Case #2: yes

Note

«Rock, Paper and Scissors» is a game which played by two person. They should play Rock, Paper or Scissors by their hands at the same time. Rock defeats scissors, scissors defeats paper and paper defeats rock. If two people play the same item, the game is tied..

Problem F. Fruit Ninja

Input file:	`standard input`
Output file:	`standard output`
Time limit:	15 seconds
Memory limit:	256 mebibytes

Fruit Ninja is a juicy action game enjoyed by millions of players around the world, with squishy, splat and satisfying fruit carnage! Become the ultimate bringer of sweet, tasty destruction with every slash. (From Wikipedia)

It is a very popular game on cell phones where people can enjoy cutting the fruit by touching the screen. The screen is rectangular, and all the fruit can be considered as circles, with coordinate of the center, and radius. Note that the fruit may overlap with each other. In this problem, a touch is a straight line cutting through the whole screen, scoring all the fruits it cuts or touches.

Now Fred is playing the Fruit Ninja, and seems absorbed in the game. He's desperate to create a new record, so he asks you for help. Now you are given a screen shot of the game, help him find the highest score he can get in a single touch.

Input

The first line contains an integer $T(1 \leqslant T \leqslant 50)$, indicating the number of test cases. Each test case contains several lines. The first line contains an integer $N(1 \leqslant N \leqslant 1000)$, indicating the number of fruit. The following N lines each contains three integers $X_i, Y_i, R_i(-1000 \leqslant X, Y \leqslant 1000, 1 \leqslant R_i \leqslant 1000)$, representing a fruit on the screen, where (X, Y) is the coordinate of the center of the fruit, and R_i is the radius. You can assume the screen is infinite.

Output

For each test case in the input, print one line: "`Case #X: Y`", where X is the test case number (starting with 1) and Y is maximum number of fruit that you can cut in a single touch.

Examples

standard input	standard output
2 4 -2 5 1 5 5 1 -3 2 1 0 1 1 4 -4 5 1 3 2 1 -5 3 1 4 -3 1	Case #1: 3 Case #2: 2

Problem G. GRE Words

Input file:	`standard input`
Output file:	`standard output`
Time limit:	5 seconds
Memory limit:	256 mebibytes

Recently George is preparing for the Graduate Record Examinations (GRE for short). Obviously the most important thing is reciting the words.

Now George is working on a word list containing N words. He has so poor a memory that it is too hard for him to remember all of the words on the list. But he does find a way to help him to remember. He finds that if a sequence of words has a property that for all pairs of neighboring words, the previous one is a substring of the next one, then the sequence of words is easy to remember.

So he decides to eliminate some words from the word list first to make the list easier for him. Meantime, he doesn't want to miss the important words. He gives each word an importance, which is represented by an integer ranging from -1000 to 1000, then he wants to know which words to eliminate to maximize the sum of the importance of remaining words. Negative importance just means that George thought it useless and is a waste of time to recite the word.

Note that although he can eliminate any number of words from the word list, he can never change the order between words. In another word, the order of words appeared on the word list is consistent with the order in the input. In addition, a word may have different meanings, so it can appear on the list more than once, and it may have different importance in each occurrence.

Input

The first line contains an integer $T(1 \leqslant T \leqslant 50)$, indicating the number of test cases. Each test case contains several lines. The first line contains an integer $N(1 \leqslant N \leqslant 2 \times 10^4)$, indicating the number of words. Then N lines follows, each contains a string S_i and an integer W_i, representing the word and its importance. S_i contains only lowercase letters. You can assume that the total length of all words will not exceeded 3×10^5.

Output

For each test case in the input, print one line: "`Case #X: Y`", where X is the test case number (starting with 1) and Y is the largest importance of the remaining sequence of words.

Examples

standard input	standard output
1 5 a 1 ab 2 abb 3 baba 5 abbab 8	Case #1: 14

Problem H. Holiday's Accomodation

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 meibibytes

Nowadays, people have many ways to save money on accommodation when they are on vacation. One of these ways is exchanging houses with other people. Here is a group of N people who want to travel around the world. They live in different cities, so they can travel to some other people's city and use someone's house temporary. Now they want to make a plan that choose a destination for each person. There are 2 rules should be satisfied:

1. All the people should go to one of the other people's city.
2. Two of them never go to the same city, because they are not willing to share a house.

They want to maximize the sum of all people's travel distance. The travel distance of a person is the distance between the city he lives in and the city he travels to. These N cities have $N-1$ highways connecting them. The travelers always choose the shortest path when traveling. Given the highways' information, it is your job to find the best plan, that maximum the total travel distance of all people.

Input

The first line of input contains one integer $T(1 \leqslant T \leqslant 10)$, indicating the number of test cases. Each test case contains several lines. The first line contains an integer $N(2 \leqslant N \leqslant 10^5)$, representing the number of cities. Then the following $N-1$ lines each contains three integers $X, Y, Z(1 \leqslant X, Y \leqslant N, 1 \leqslant Z \leqslant 10^6)$, means that there is a highway between city X and city Y, and length of that highway. You can assume all the cities are connected and the highways are bi-directional.

Output

For each test case in the input, print one line: "`Case #X: Y`", where X is the test case number (starting with 1) and Y represents the largest total travel distance of all people.

Examples

standard input	standard output
2 4 1 2 3 2 3 2 4 3 2 6 1 2 3 2 3 4 2 4 1 4 5 8 5 6 5	Case #1: 18 Case #2: 62

Problem I. Isabella's Message

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 mebibytes

Isabella and Steve are very good friends, and they often write letters to each other. They exchange funny experiences, talk about people around, share their feelings and write about almost everything through the letters. When the letters are delivered, they are quite afraid that some other people(maybe their parents) would peek. So they encrypted the letter, and only they know how to decrypt it. This guarantees their privacy. The encrypted message is an $N \times N$ matrix, and each grid contains a character. Steve uses a special mask to work as a key. The mask is $N \times N$(where N is an even number) matrix with $\frac{N \times N}{4}$ holes of size 1×1 on it. The decrypt process consist of the following steps:

1. Put the mask on the encrypted message matrix
2. Write down the characters you can see through the holes, from top to down, then from left to right.
3. Rotate the mask by 90 degrees clockwise.
4. Go to step 2, unless you have wrote down all the $N \times N$ characters in the message matrix.
5. Erase all the redundant white spaces in the message.

For example, you got a message shown in figure 1, and you have a mask looks like figure 2. The decryption process is shown in figure 3, and finally you may get a message "`good morning`".

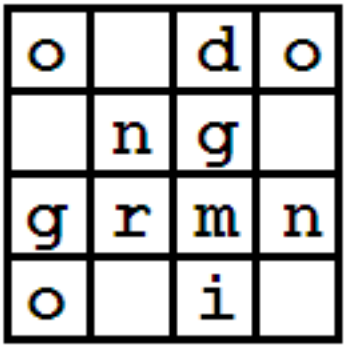

figure.1

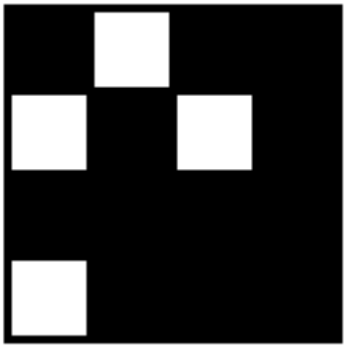

figure.2

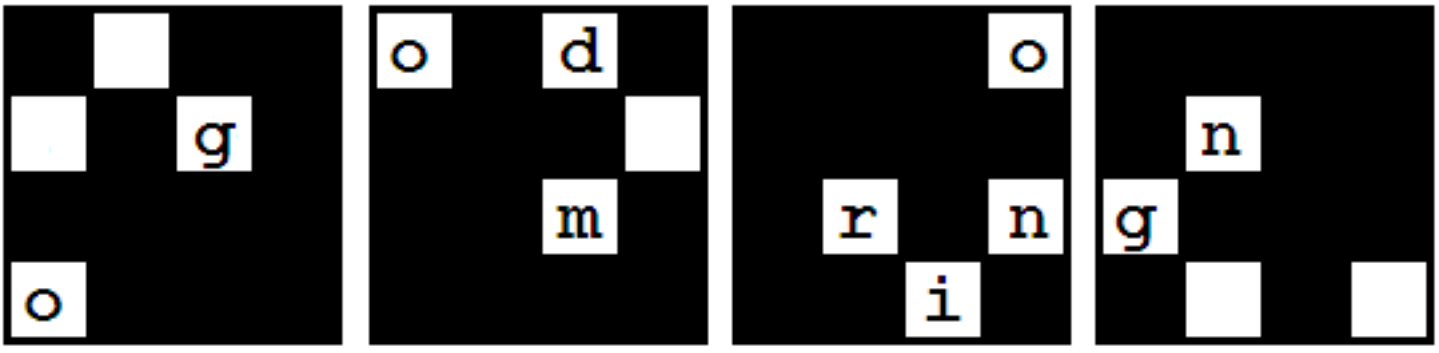

figure.3

You can assume that the mask is always carefully chosen that each character in the encrypted message will appear exactly once during decryption. However, in the first step of decryption, there are several ways to put the mask on the message matrix, because the mask can be rotated (but not flipped). So you may get different results such as "`od morning go`" (as showed in figure 4), and you may also get other messages like "`orning good m`", "`ng good morni`".

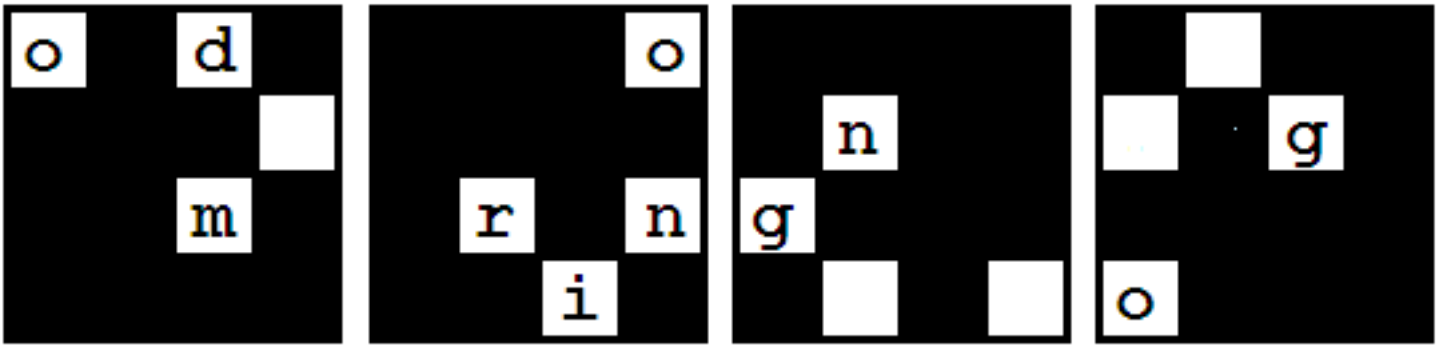

figure.4

Steve didn't know which direction of the mask should be chosen at the beginning, but after he tried all possibilities, he found that the message "`good morning`" is the only one he wanted because he couldn't recognize some words in the other messages. So he will always consider the message he can understand the correct one. Whether he can understand a message depends whether he knows all the words in the message. If there are more than one ways to decrypt the message into an understandable one, he will choose the lexicographically smallest one. The way to compare two messages is to compare the words of two messages one by one, and the first pair of different words in the two messages will determine the lexicographic order of them. Isabella sends letters to Steve almost every day. As decrypting Isabella's message takes a lot of time, and Steve can wait no longer to know the content of the message, he asked you for help. Now you are given the message he received, the mask, and the list of words he already knew, can you write a program to help him decrypt it?

Input

The first line contains an integer $T(1 \leqslant T \leqslant 100)$, indicating the number of test cases. Each test case contains several lines. The first line contains an even integer $N(2 \leqslant N \leqslant 50)$, indicating the size of the matrix. The following N lines each contains exactly N characters, reresenting the message matrix. The message only contains lowercase letters and periods('.'), where periods represent the white spaces. You can assume the matrix contains at least one letter. The following N lines each contains N characters, representing the mask matrix. The asterisk('*') represents a hole, and period('.') otherwise. The next line contains an integer $M(1 \leqslant M \leqslant 100)$, the number of words he knew. Then the following M lines each contains a string represents a word. The words only contain lowercase letters, and its length will not exceed 20.

Output

For each test case in the input, print one line: "`Case #X: Y`", where X is the test case number (starting with 1) and Y is Isabella's message. If Steve cannot understand the message, just print the Y as "`FAIL TO DECRYPT`".

Examples

standard input	standard output
3 4 o.do .ng. grmn o.i. .*.. *.*. .... *... 2 good morning 4 ..lf eoyv oeou vrer ..*. .*.. .... *.*. 5 i you the love forever 4 .sle s.c. e.fs ..uu *... .*.. ...* ..*. 1 successful	Case #1: good morning Case #2: love you forever Case #3: FAIL TO DECRYPT

Problem J. Ji-Tu Problem

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 mebibytes

There are some chickens and rabbits in the cage. They have fifteen heads and forty feet in all. How many chickens and rabbits are there respectively? It is a classical math problem which can date back to the Northern and Southern Dynasties (420-589). Here is an interesting algorithm to solve the problem: Assume that the chickens and rabbits are well trained. You whistle, and all of them lift a leg, then there are $40 - 15 = 25$ feet on the floor. You whistle again, and there are $25 - 15 = 10$ legs remain standing. After two whistles, all the chickens sit on the floor, and all the rabbits stand on two legs. So there are $10/2 = 5$ rabbits and $15 - 5 = 10$ chickens.

John has a farm with lots of animals in it. He is now facing the similar problem. There are **exactly** N kinds of animals and he wants to know their quantities. He only knows that different kinds of animals have different number of legs (at least one), but he has no idea how many legs they each have. He trains the animals and tries to figure it out using the algorithm stated above. First he makes all the animals stand up with all their legs and counts their legs. Then, for each time he whistles, all the animals lift one leg(if it has at least one leg standing on the ground), and then he counts the feet again. After K times, he thinks that it is enough to determine the quantity of each kind of animal, but does it really work? So, it is your job to help him to solve the problem.

Input

The first line contains an integer $T(1 \leqslant T \leqslant 100)$, indicating the number of test cases. Each test case contains two lines. The first line contains two integers $N(1 \leqslant N \leqslant 1000)$ and $K(1 \leqslant K \leqslant 1000)$, representing the number of different kinds of animals and the time he whistles. The second line contains $K + 1$ integers $A_0, A_1 \ldots A_K (0 \leqslant A_i \leqslant 10^4)$ where A_i represents the number of legs after his i^{th} whistle.

Output

For each test case in the input, print several lines. The first line contains "`Case #X:`", where X is the test case number (starting with 1). The next line contains "`No Solution`", "`Unique Solution`" or "`Multiple Solutions`" according to the result.

If the result is uniquely determined, you should print N extra lines each contains two integer L_i, N_i, where L_i represents how many legs does the i^{th} kind of animal have and N_i represents the number of i^{th} kind of animal. The animals should be sorted by the number of their legs in ascending order.

Examples

standard input	standard output
3 2 3 14 9 6 3 2 2 8 5 3 3 2 20 13 8	Case #1: Unique Solution 1 2 4 3 Case #2: No Solution Case #3: Multiple Solutions