Problem A

Cluster Analysis

Cluster analysis, or also known as clustering, is a task to group a set of objects into one or more groups such that objects belong to a same group are more similar compared to object in other groups. In this problem, you are given a set of N positive integers and an integer K. Your task is to compute how many clusters are there in the given set where two integers belong to a same cluster if their difference is no larger than K.

For example, let there be a set of N = 7 positive integers: 2, 6, 1, 7, 3, 4, 9, and K = 1.

Based-on the cluster definition of K, we know that:

- 2 and 1 belong to a same cluster (the difference is no more than K = 1),
- 2 and 3 belong to a same cluster,
- 6 and 7 belong to a same cluster,
- 3 and 4 belong to a same cluster.

From these observations, we can conclude that there are 3 clusters in this example: {2, 1, 3, 4}, {6, 7}, and {9}.

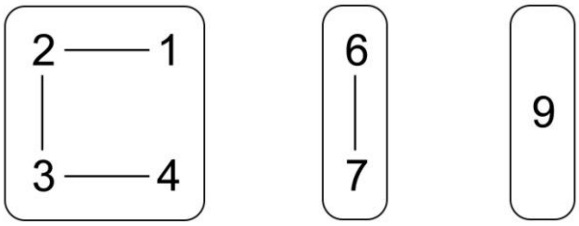


Figure 1.

Figure 1 illustrates the clustering result. A line connecting two numbers means that those two numbers should belong to a same cluster according to the definition.

Input

The first line of input contains an integer T ($T \leq 100$) denoting the number of cases. Each case begins with two integers in a line: N and K ($1 \leq N \leq 100$; $1 \leq K \leq 1{,}000{,}000$) denoting the set size and the clustering parameter respectively. The next line contains N integers A_i ($1 \leq A_i \leq 1{,}000{,}000$) representing the set of positive integers. You are guaranteed that all integers in the set are unique.

Output

For each case, output "`Case #X: Y`", where `X` is the case number starts from 1 and `Y` is the number of cluster for that particular case.

Sample Input	Output for Sample Input
4 7 1 2 6 1 7 3 4 9 7 2 2 6 1 7 3 4 9 5 5 15 1 20 4 17 8 10 100 200 300 400 500 600 700 800	Case #1: 3 Case #2: 1 Case #3: 2 Case #4: 8

Explanation for 2nd sample case

The given set is exactly the same as in 1st sample, however, now K = 2. With two additional observations (compared to 1st sample): 4 and 6 are in the same cluster, 7 and 9 are in the same cluster; all those integers will be in the same cluster.

Explanation for 3rd sample case

There are 2 clusters: {1, 4}, and {15, 20, 17}.

Explanation for 4th sample case

In this sample, all integers will be in their own cluster.

Problem B

Body Building

Bowo is fed up with his body shape. He has a tall posture, but he's very skinny. No matter how much he eats, he never gains any weight. Even though he is a computer geek (and he loves it), he wants a pretty and non-geek girlfriend. Unfortunately, most girls in his surrounding do not like skinny and unattractive guy. Therefore, Bowo has decided to gain some muscles in his body; he joined a fitness club and begun to do some body building exercises.

There are a lot of exercise equipments in a fitness club, and usually there should be weightlifting equipments such as barbell and dumbbell (barbell with shorter rod). Upon seeing a dumbbell, Bowo cannot help but imagining graphs which are similar to a dumbbell. A graph – which later referred as "connected component" – of N nodes is called a dumbbell if it fulfills all the following conditions:

(i) All nodes in the graph can be partitioned into two disjoint sets P and Q which have equal size, i.e. N / 2 nodes each.
(ii) Both induced subgraph of P and Q are complete graphs.
(iii) P and Q are connected by exactly one edge.

Informally, a dumbbell is obtained by connecting two equal size complete graphs with an edge.

For example, consider graph A in Figure 1 with 10 nodes and 21 edges. There are two disjoint complete graphs of size 5 which are connected by an edge. Therefore, this graph is a dumbbell. Graph B and C are also dumbbells. Graph D, on the other hand, is not.

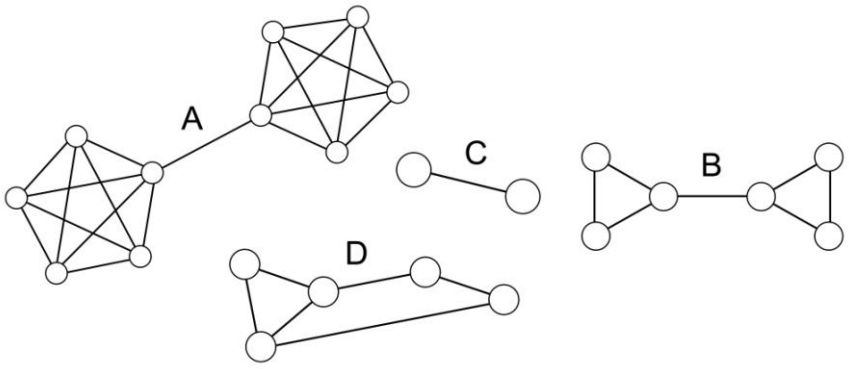


Figure 1.

Given a graph (which might be disconnected), determine how many connected components which are dumbbells. A connected component is a connected subgraph which no vertex can be added and still be connected.

Input

The first line of input contains an integer T ($T \leq 50$) denoting the number of cases. Each case begins with two integers: N and M ($1 \leq N \leq 100$; $0 \leq M \leq 4{,}950$) denoting the number of nodes and edges in the graph respectively. The nodes are numbered from 1 to N. The following M lines each contains two integer: a and b ($1 \leq a, b \leq N$; $a \neq b$) representing an undirected edge connecting node a and node b. You are guaranteed that each pair of nodes has at most one edge in the graph.

Output

For each case, output "`Case #X: Y`", where `X` is the case number starts from 1 and `Y` is the number of connected components which are dumbbells for the respective case.

Sample Input	Output for Sample Input
4 1 0 4 2 1 2 3 4 10 10 1 2 1 3 2 3 3 4 4 5 5 6 4 6 7 8 8 9 9 10 9 5 1 2 3 4 5 6 7 8 8 9	Case #1: 0 Case #2: 2 Case #3: 2 Case #4: 3

Explanation for 1st sample case

There is only one node in the graph; a dumbbell requires at least two nodes.

Explanation for 2nd sample case

Both connected components are dumbbells: {1, 2} and {3, 4}.

Explanation for 3rd sample case

There are two connected components: {1, 2, 3, 4, 5, 6}, and {7, 8, 9, 10}, and both of them are dumbbells. The first one is dumbbell with complete graph of size 3, while the second one has size of 2.

Explanation for 4th sample case

There are four connected components: {1, 2}, {3, 4}, {5, 6} and {7, 8, 9}. Only the first three are dumbbells.

Problem C

Electric Bike

Two years ago, Putri bought an electric bike (e-bike). She likes e-bike a lot since it can assist her in cycling through steep roads when commuting to work. Over time, the battery capacity decreases. Now, her e-bike battery capacity is so low that she needs to carefully plan when to turn on the assist and at which level; different level of assist consumes different amount of energy and produces different assist power.

Putri divides the road that she travels from her home to her workplace into N segments where each segment has a steepness level. For example, the figure below shows a road with N = 7 segments.

Segment	#1	#2	#3	#4	#5	#6	#7
Steepness	10	3	0	1	2	15	9

From her home, Putri has to travel from the first road segment, to the second road segment, and so on, until the last segment to reach her work place. In the example above, the first segment has steepness of 10 which means Putri has to pedal her bike with 10 unit of energy. Her e-bike has fixed 4 assist levels where each level generates different power, as shown in the following table:

Assist Level	0	1	2	3
Assist Power	0	4	8	11

Assist level L will consume L unit of energy from the battery and will give Putri additional pedaling assist power according to the table above. Putri can only change the assist level to level L if the battery has at least L energy left and she is at the beginning of a road segment. Setting an assist level where the generated power is higher than the road steepness will cause the excess energy power to be wasted.

For example, if Putri sets the assist level L = 2 at the beginning of the first road segment (with steepness 10), then she only needs to pedal her bike with 2 unit of energy instead of 10 (since her e-bike is assisting her with 8 unit of energy) to advance to the second road segment. If Putri sets the assist level L = 3, her e-bike generates more power to handle the steepness 10, thus she does not need to pedal at all, and the excess energy is wasted.

Putri can change the assist level instantly before entering a road segment, however, she does not want to change the assist level more than K times (it's too tiring). If there is not enough energy in the battery to support the selected assist level for the road segment, the e-bike will shutdown at the beginning of the road segment and Putri has to pedal through the rest of the road segments, i.e. the assist level automatically set to 0 for the rest of the journey. Note that Putri can change her e-bike assist level (given it's still less than K) at the beginning of the road segment to avoid shutdown by force. Initially at her home, the assist level is set to 0 and the battery is fully charged with E unit of energy. Putri wants to know the minimum energy she will need to pedal the bike to reach the workplace if she utilizes her e-bike optimally.

Input

The first line of input contains T ($T \leq 100$) denoting the number of cases. Each case begins with three integers: N, K, and E in a line ($1 \leq N \leq 1{,}000$; $0 \leq K \leq 10$; $0 \leq E \leq 50$) as described in the problem statement above. The next line contains N non-negative integers denoting the steepness level of ith segment where i = 1..N respectively. The steepness level of any road segment is at most 15.

Output

For each case, output "`Case #X: Y`", where `X` is the case number starts from 1 and `Y` is the minimum energy Putri needs to pedal the e-bike from her home to her workplace.

Sample Input	Output for Sample Input
5 7 2 10 10 3 0 1 2 15 9 7 2 15 10 3 0 1 2 15 9 7 3 15 10 3 0 1 2 15 9 5 2 5 11 15 1 14 12 15 8 30 2 14 6 1 2 13 14 12 13 12 7 12 1 2 10	Case #1: 14 Case #2: 8 Case #3: 4 Case #4: 34 Case #5: 18

Explanation for 1st sample case

Putri changes the assist level to 1 at (the beginning of) road segment #2, then change the assist level to 3 at road segment #6. Thus, she needs to pedal with 10 unit of energy for road segment #1 and 4 unit of energy for road segment #6. Note that if she changes the assist level to 1 at road segment #1 and then to assist level 3 at road segment #6, then at the beginning of road segment #7 the battery only has 2 unit of energy left and will automatically shutdown to assist level 0, thus Putri has to pedal with 9 energy for road segment #7.

Explanation for 2nd sample case

Putri changes the assist level to 3 at road segment #1 and then changes to assist level 2 at road segment #2. Thus, she only needs to pedal 7 unit of energy for road segment #6 and 1 unit of energy for road segment #7.

Explanation for 3rd sample case

Putri changes the assist level to 3 at road segment #1, then changes to assist level 1 at road segment #2, finally changes to assist level 3 at road segment #6. Thus, she only needs to pedal 4 unit of energy for road segment #6.

Problem D

Kevin's Problem

In his applied probability class, Kevin learned about the Secretary Problem.

> There are N applicants applying for secretary position in a company. The applicants are interviewed by a hiring manager one by one in arbitrary order. During the interview, the manager can rank the applicant's quality uniquely relative to all applicants which have been interviewed so far, but he has no idea of the quality of applicants yet to be interviewed. The decision whether the manager should hire or reject the applicant must be done right after the interview ended (before he starts an interview with other applicant) and cannot be changed, i.e. once rejected, that particular applicant cannot be considered anymore. There is only one position available, so what is the best decision strategy for this problem?

One reasonable strategy is: reject the first K applicants and hire the first remaining applicant who is better than all previous interviewed applicants, or hire the last one if there is no such applicant.

Unfortunately, Kevin did not pay a full attention in his class. He misunderstood the strategy; instead of "… hire the first remaining applicant who is better than all previous interviewed applicants …", he thought it is "… hire the first remaining applicant who is better than the (immediate) previous interviewed applicant …". Let's call this variation as Kevin's strategy.

Given N, K, and p determine in how many ways (interview order) such that the applicant whose rank is p among all applicants will be selected by Kevin's strategy. For example, let N = 4, K = 2, and p = 2. Among all permutation of 1..N, there are only 7 possible permutation such that the 2nd rank applicant is selected by Kevin's strategy.

- 1, 3, 2, 4 – 2 is selected because it's better than 3.
- 1, 3, 4, 2 – 2 is selected because 4 is worse than 3, and 2 is better than 4.
- 1, 4, 2, 3
- 3, 1, 4, 2
- 3, 4, 2, 1
- 4, 1, 3, 2
- 4, 3, 2, 1

Note that the first 2 applicants will not be hired in this example (K = 2). Clearly, the 2nd rank applicant will not be selected in any permutation where she appears in the first K.

An applicant has a better rank if her rank is higher (smaller number), e.g. 2nd rank is better than 5th.

Input

The first line of input contains an integer T ($T \le 100$) denoting the number of cases. Each case contains three integers: N, K, and p ($2 \le N \le 500$; $1 \le K < N$; $1 \le p \le N$) as explained in the problem description above.

Output

For each case, output "`Case #X: Y`", where `X` is the case number starts from 1 and `Y` is the number of permutation of 1… N such that the p-th rank applicants is selected by Kevin's strategy for that particular case. As this number could be very large, modulo `Y` by 1,000,000,007.

Sample Input	Output for Sample Input
4 4 2 2 5 2 3 5 3 5 8 4 2	Case #1: 7 Case #2: 26 Case #3: 12 Case #4: 7890

Explanation for 3rd sample case

There are 5 applicants and Kevin's strategy will reject the first 3. 5th rank applicant will be selected only if she is interviewed last, thus the manager has no choice but to hire her. Among 24 possible permutations where 5th rank applicant appears last, only 12 permutations which make her hired:

1, 2, 3, 4, 5	2, 1, 3, 4, 5	3, 1, 2, 4, 5	4, 1, 2, 3, 5
1, 3, 2, 4, 5	2, 3, 1, 4, 5	3, 2, 1, 4, 5	4, 2, 1, 3, 5
1, 4, 2, 3, 5	2, 4, 1, 3, 5	3, 4, 1, 2, 5	4, 3, 1, 2, 5

Some examples where 5th rank will not be hired even though she is the last:
1, 2, 4, 3, 5 -- 3rd rank will be hired.
1, 4, 3, 2, 5 -- 2nd rank will be hired.
3, 2, 4, 1, 5 -- 1st rank will be hired.
…

Problem E

Cutting Tree

Tree in graph theory refers to any connected graph (of nodes and edges) which has no simple cycle, while forest corresponds to a collection of one or more trees. In this problem, you are given a forest of N nodes (of rooted trees) and K queries. Each query is in the form of:

- `C` x : remove the edge connecting node x and its parent.
  If node x has no parent, then ignore this query.
- `Q` a b : output YES if there is a path from node a to node b in the forest; otherwise, NO.

For example, let the initial forest is shown by Figure 1.

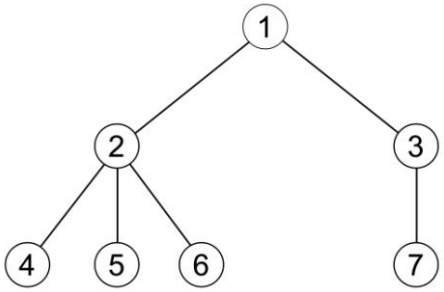


Figure 1.

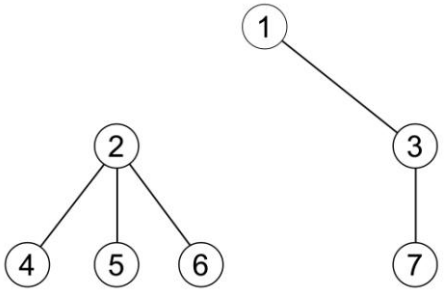


Figure 2.

Let's consider the following queries (in order):

1) `Q 5 7` : output YES.
2) `C 2` : remove edge (2, 1) – the resulting forest is shown in Figure 2.
3) `Q 5 7` : output NO, as there is no path from node 5 to node 7 in Figure 2.
4) `Q 4 6` : output YES.

Input

The first line of input contains an integer T ($T \le 50$) denoting the number of cases. Each case begins with two integers: N and K ($1 \le N \le 20{,}000$; $1 \le K \le 5{,}000$) denoting the number of nodes in the forest and the number of queries respectively. The nodes are numbered from 1 to N. The next line contains N integers P_i ($0 \le P_i \le N$) denoting the parent of ith node respectively. $P_i = 0$ means that node i does not have any parent (i.e. it's a root of a tree). You are guaranteed that the given input corresponds to a valid forest. The next K lines represent the queries. Each query is in the form of "`C` x" or "`Q` a b" ($1 \le x, a, b \le N$), as described in the problem statement above.

Output

For each case, output "`Case #X:`" in a line, where `X` is the case number starts from 1. For each "`Q` a b" query in the input, output either "`YES`" or "`NO`" (without quotes) in a line whether there is a path from node a to node b in the forest.

Sample Input	Output for Sample Input
4 7 4 0 1 1 2 2 2 3 Q 5 7 C 2 Q 5 7 Q 4 6 4 4 2 0 2 3 C 3 Q 1 2 C 1 Q 1 2 3 5 0 3 0 C 1 Q 1 2 C 3 C 1 Q 2 3 1 1 0 Q 1 1	Case #1: YES NO YES Case #2: YES NO Case #3: NO YES Case #4: YES

Explanation for 2nd sample case

The initial forest is shown in Figure 3 below.

1) `C 3` : remove edge (3, 2) – the resulting forest is shown in Figure 4.
2) `Q 1 2` : output YES.
3) `C 1` : remove edge (1, 2) – the resulting forest is shown in Figure 5.
4) `Q 1 2` : output NO as there is no path from node 1 to node 2 in Figure 5.

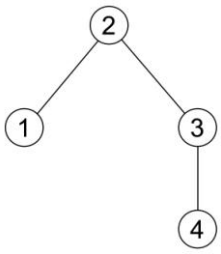


Figure 3.

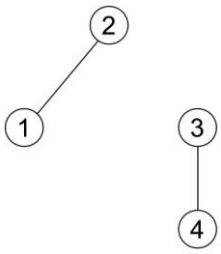


Figure 4.

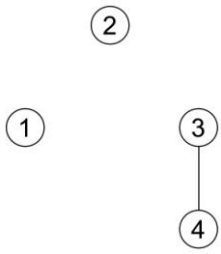


Figure 5.

Problem F

Double Swords

Last night, Kingdom of Light was attacked by Kingdom of Dark! The queen of Kingdom of Light, Queen Ar, was captured and locked inside a dark and creepy castle. The king of Kingdom of Light, King Ash, wants to save the queen.

The castle is guarded by N dragons, conveniently numbered from 1 to N. To save Queen Ar, King Ash must kill all the dragons. The kingdom's oracle said that in order to kill the ith dragon, King Ash has to slay it with exactly two swords, one in each hand: one sword of length A_i units, and another sword of length between B_i and C_i, inclusive. King Ash can carry unlimited number of swords, and each sword can be used multiple times.

The number of blacksmiths in the kingdom is limited, so it is important to make as few swords as possible. Besides, making swords is expensive and takes much time. Determine the minimum number of swords the kingdom has to make so that King Ash can save Queen Ar!

Input

The first line of input contains an integer T ($T \le 20$) denoting the number of cases. Each case begins with an integer N ($1 \le N \le 100{,}000$) denoting the number of dragons which have to be killed. The next N lines each contains three integers: A_i, B_i, and C_i ($1 \le A_i \le 1{,}000{,}000$; $1 \le B_i \le C_i \le 1{,}000{,}000$) denoting the swords' length needed to kill the ith dragon as described in the above problem statement.

Output

For each case, output "`Case #X: Y`", where `X` is the case number starts from 1 and `Y` is the minimum number of swords the kingdom has to make and carry in order to defeat all the dragons and save Queen Ar.

Warning: large input/output data, be careful with certain input-output routines.

Sample Input	Output for Sample Input
4 1 7 6 8 3 3 5 10 6 11 15 3 13 15 4 1 10 20 3 50 60 2 30 40 4 70 80 2 5 100 200 150 1000 2000	Case #1: 2 Case #2: 3 Case #3: 8 Case #4: 3

Explanation for 1st sample case
The kingdom has to make two swords in order to defeat one dragon.

Explanation for 2st sample case
All the dragons can be defeated by three swords, for example, with lengths of: 3, 6, and 15.

- The fist dragon can be defeated by swords of length 3 and 6.
- The second dragon can be defeated by swords of length 6 and 15.
- The third dragon can be defeated by swords of length 3 and 15.

There also exists other combination of three swords.

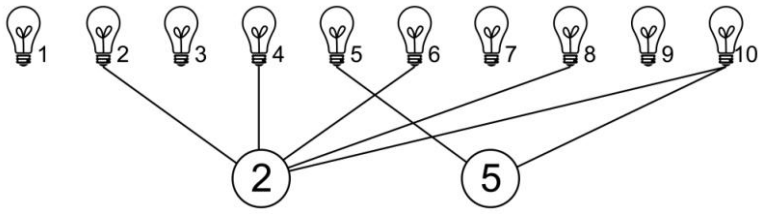
Problem G

Prime Switch

There are N lamps (uniquely numbered from 1 to N) and K switches. Each switch has one prime number written on it and it is connected to all lamps whose number is a multiple of that prime number. Pressing a switch will toggle the condition of all lamps which are connected to the pressed switch; if the lamp is off then it will be on, and vice versa. You can press only one switch at one time; in other words, no two switches can be pressed together at the same time. If you want to press multiple switches, you should do it one by one, i.e. allowing the affected lamps of the previous switch toggle their condition first before pressing another switch.

Initially all the lamps are off. Your task is to determine the maximum number of lamps which can be turned on by pressing one or more switches.

For example, let there be 10 lamps (1…10) and 2 switches which numbers are 2 and 5 as shown in the following figure.



In this example:

- Pressing switch 2 will turn on 5 lamps: 2, 4, 6, 8, and 10.
- Pressing switch 5 will turn on 2 lamps: 5 and 10.
- Pressing switch 2 and 5 will turn on 5 lamps: 2, 4, 5, 6, and 8. Note that lamp number 10 will be turned off as it is toggled twice, by switch 2 and switch 5 (off → on → off).

Among all possible switches combinations, the maximum number of lamps which can be turned on in this example is 5.

Input

The first line of input contains an integer T ($T \leq 100$) denoting the number of cases. Each case begins with two integers in a line: N and K ($1 \leq K \leq N \leq 1{,}000$), denoting the number of lamps and switches respectively. The next line contains K distinct prime numbers, each separated by a single space, representing the switches number. You are guaranteed that the largest number among those switches is no larger than N.

Output

For each case, output "`Case #X: Y`", where `X` is the case number starts from 1 and `Y` is the maximum number of lamps which can be turned on for that particular case.

Sample Input	Output for Sample Input
4 10 2 2 5 21 4 2 3 5 7 100 1 5 100 3 3 19 7	Case #1: 5 Case #2: 11 Case #3: 20 Case #4: 42

Explanation for 2nd sample case
You should press switch 2 and 7, such that 11 lamps will be turned on: 2, 4, 6, 7, 8, 10, 12, 16, 18, 20, and 21. There exist some other combinations which can turn on 11 lamps, but none can turn more than 11 lamps on.

Explanation for 3rd sample case
There is only one switch, and pressing it will turn 20 lamps on.

Explanation for 4th sample case
Pressing all switches will turn 42 lamps on, and it is the maximum possible in this case.

Problem H

I Want That Cake

There was an interesting game played on a popular Korean reality TV Show. 12 players in 3 teams – 4 persons in each team – lined up in one row in random order. The game master approaches the players one by one starting from the most front player, with a tray full of 31 cakes. Each player should eat at least 1 and at most 5 cakes during his/her turn. The player who eats the last cake will lose along with his/her group (this is a team game). It was more an entertainment show rather than a real competition. The players were selected from "chubby" celebrities who like to eat, thus causing them in dilemma whether they should win the game (which might require them to hold their urge to eat the cakes), or just enjoy all 5 cakes ignoring their team member.

This problem is related to the game. There are 2 teams (A and B) each with N players lined up in a row in random order. Initially there are M cakes in the tray, and each player (starting from the most front) has to eat at least 1 and at most K cakes during his/her turn. The team whose player eat the last cake win (note that this is different compared to the original game).

Your task is to determine which team will win the game, given both teams play optimally.

Input

The first line of input contains an integer T ($T \le 100$) denoting the number of cases. Each case begins with three integers N, M, and K ($1 \le N \le 1{,}000$; $1 \le K \le M \le 2*N$) in a line denoting the number of players in each team, the initial number of cakes, and the maximum number of cakes can be eaten by each player in his/her turn respectively. The next line contains a string S representing the players order from the front most to the last player. S consists of only character 'A' or 'B' representing which team does the respective player belongs to. The length of S is exactly $2*N$ and the number of 'A' and 'B' will be equal.

Output

For each case, output "`Case #X: Y`", where `X` is the case number starts from 1 and `Y` is 'A' if the game will be won by team A, otherwise 'B'. Assume both teams will play the game optimally to win the game.

Sample Input	Output for Sample Input
4 3 5 2 AAABBB 4 7 2 AAABBBBA 4 5 2 BABABABA 4 6 3 BAABBABA	Case #1: A Case #2: B Case #3: B Case #4: A

Explanation for 1st sample case
The first two players of team A each needs to eat 2 cakes, while the third A player eats the last cake.

Explanation for 2nd sample case
No matter what team A do, the last cake will be eaten by one of team B's player.

Explanation for 3rd sample case
To ensure their win, the first player (B) should eat 2 cakes, leaving only 3 cakes to the next player (A). This player (A) should eat at least 1 and at most 2 cakes. No matter how many cakes this player eats, the next player (B) will eat the last cake.

Problem I

Maze Mayhem

As the title implies, your task in this problem is related to maze, specifically a 2D maze of size N x M (N rows and M columns). Starting from the top-left most of the maze – coordinate (1, 1), you should clear the maze by moving to the goal point at the bottom-right most of the maze – coordinate (N, M). To simplify things, you are only allowed to move right or down at any step. Of course like any other mazes, there might be cells with obstacle which cannot be visited. For example, consider the Figure 1 below. There are two obstacles, one at (3, 2) and the other at (2, 4). The black line with arrow is one possible path from the starting point to the goal.

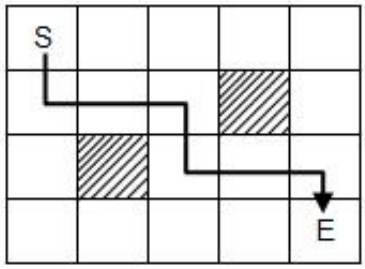


Figure 1.

Problems related to such maze are commonly found in programming contest. Usually for such problem, preparing the test data (i.e. the input maze) is much more troublesome than solving the problem itself. If we generate a random maze, then chances are the mazes will be invalid. A maze is invalid if at least one of the following conditions applies:

- There is an obstacle at the starting point (1, 1) and/or at the goal point (N, M).
- There is no valid path from the starting point to the goal point.

A path is valid if it consists of only moving right or down, i.e. from (r, c) you can only go to (r + 1, c) or (r, c + 1), and there is no obstacle along the path (visited cells).

Given the size of the maze (N x M) and an integer K, determine how many invalid mazes of such size which have at most K obstacles. Output the result modulo 1,000,000,007.

Input

The first line of input contains an integer T ($T \leq 100$) denoting the number of cases. Each case contains three integers: N, M, and K ($1 \leq N, M \leq 8$; $N * M > 1$; $0 \leq K \leq N * M$) denoting the size of the maze and the maximum number of obstacles respectively.

Output

For each case, output "`Case #X: Y`", where `X` is the case number starts from 1 and `Y` is the number of invalid mazes with the given size and maximum number of obstacles, modulo by 1,000,000,007.

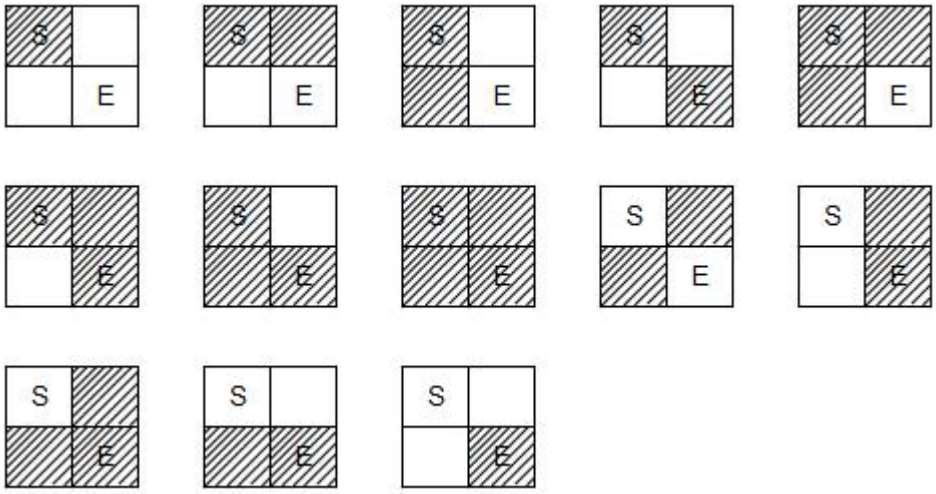
Sample Input	Output for Sample Input
4 1 2 2 2 2 4 2 2 2 4 3 11	Case #1: 3 Case #2: 13 Case #3: 8 Case #4: 3800

Explanation for 1st sample case

Out of 4 possible mazes, only 1 which is valid (no obstacle in both cells). Thus there are 3 mazes which are invalid.

Explanation for 2nd sample case

These are the 13 invalid mazes of size 2 x 2.



Explanation for 3rd sample case

The maze size is exactly equal to the 2nd sample, but in this case you cannot have more than 2 obstacles. There are 8 such mazes; consult figures in 2nd sample case explanation above to figure out which.

Problem J

Leveling Ground

It is important to first level the ground before you build anything on top of it (e.g., new house, shed, swimming pool, driveway, etc.), especially when there are hills on the land, otherwise whatever you built will not be stable. In case you don't understand, "leveling the ground" means making the ground flat and even (having the same height). In this problem, you are given a land description and the length of land – M – that you want to level; your task is to determine the minimum amount of land you should dispose in order to have a level land of length M. Note that in this problem you are only allowed to dispose land, not filling it.

The total length of the given land will be N, and the land will be encoded with the following format:

(1) / means ascending slope (disposing an ascending slope cost 0.5),
(2) \ means descending slope (disposing a descending slope cost 0.5),
(3) _ means flat (disposing a flat land cost 0),
(4) . means full land (disposing a full land cost 1).

Note that the input will only describe the land's surface, thus (4) will not appear in any input. Also note that (1) and (2) are not level.

For example, consider the following input.

```
Input  :  //__\_//\_\\\_/___\///__\\___/_
```

The input corresponds to the following land (which length is 31).

```
                 __    /\_             __
                /..\_/...\           /..\     _
Land   :       /..........\   ___   /....\___/.
               ............\_/...\/............
               ................................
Index  :       12345678901234567890123456789 01
```

Supposed we want to level a land of length M = 7, and for some reasons, we choose the land we want to level to be at index [11, 17]. Recall that you are only allowed to dispose land, thus if you want to level the land at [11, 17], you should level it such that the height is equal to the height of land at index 14 (because it is the lowest point). In the following figure, * (stars) mark the land which should be disposed.

```
                 __    /\_             __                __    /\_             __
                /..\_/...*           /..\     _          /..\_/...|           /..\     _
               /.........**   **_   /....\___/.         /.........|      _   /....\___/.
               .........*******.\/............         .........|_____|.\/............
               ................................         ................................
Index  :       12345678901234567890123456789 01         12345678901234567890123456789 01
```

If you observe, there are 12 stars in the left figure, they are:

- 1 ascending slope (at index: 15),
- 3 descending slopes (at indexes: 11, 12, and 13),
- 3 flat lands (at indexes: 14, 16, and 17), and
- 5 full lands (2 at index 11, 1 at index 12, 1 at index 16, and another 1 at index 17).

Therefore, the cost of leveling [11, 17] is: 1 * 0.5 + 3 * 0.5 + 3 * 0 + 5 * 1 = 7.

In this example, [11, 17] is not the best choice, you can do better.

Input

The first line of input contains T ($T \leq 50$) denoting the number of cases. Each case begins with two integers N and M ($1 \leq M \leq N \leq 1{,}000{,}000$) denoting the total length of the land and the length of the land which should be leveled respectively. The following line contains a string of length N describing the land's surface. The string will only contain character '/', '\', or '_', as described in the problem statement.

Output

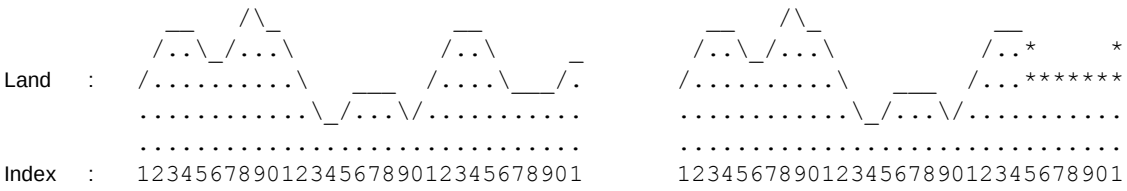
For each case, output "`Case #X: Y`", where `X` is the case number starts from 1 and `Y` is the minimum amount of land which should be disposed to achieve a level land which length is M for that particular case. Output this number with exactly one digit after the decimal point.

Warning: large input/output data, be careful with certain input-output routines.

Sample Input	Output for Sample Input
4 31 7 //___//__/___\///_____/_ 10 4 //_____\\/ 12 4 _//_/_\ 12 1 /\/\/\/\/\/\	Case #1: 3.5 Case #2: 0.0 Case #3: 1.0 Case #4: 0.5

Explanation for 1st sample case

This is the same case as the example in the problem statement. The minimum amount of land which you should dispose is 3.5. You can achieve this by leveling lands at [25, 31].



You will dispose: 1 ascending slope (at index 30), 2 descending slopes (at index 15 and 16), 4 flat lands (at index 27, 28, 29, and 31), and 2 full lands (at index 15 and 31). Therefore the total cost will be: 1 * 0.5 + 2 * 0.5 + 4 * 0 + 2 * 1 = 3.5.

Explanation for 2nd sample case

If you level the land at [3, 6] or [4, 7], you don't need to dispose any land as they are already level (have the same height).

Explanation for 3rd sample case

Level the land at [8, 11], and you only need to dispose 1 ascending slope and 1 descending slope.

Problem K

Punching Robot

In this problem, you are given a grid map of N x M (N rows and M columns) where the rows are numbered 1.. N from top to bottom, and the columns are numbered 1.. M from left to right. Your task is to count in how many ways you can reach cell (N, M) from cell (1, 1) given that you are only allowed to move right or downward at any time, i.e. if your current location is at cell (r, c), then you can only move to cell (r + 1, c) or (r, c + 1). However, we quickly realized that this kind of problem could be too easy for you, thus, not challenging. Therefore, we decided to put K punching robots in the map. Each punching robot is able to punch any object which lies in any of 3 x 3 cells centered at the robot (Figure 1). To simplify the problem, you may assume that the punching areas of any robot do not overlap.

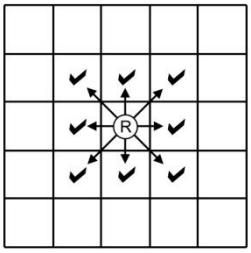


Figure 1.

Your (new) task is: count in how many ways you can reach cell (N, M) from cell (1, 1) without being punched by any robot, given that you are only allowed to move right or downward at any time. As the output can be very large, you need to modulo the output by 997. For example, consider the following map of 4 x 10 with two punching robots at (3, 3) and (2, 8).

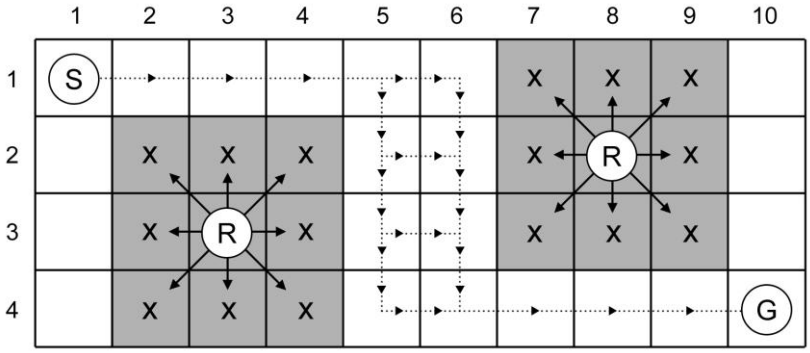


Figure 2.

In this example, there are 4 ways to reach (4, 10) from (1, 1) without being punched by any of the robots. All those 4 paths only differ when they go from (1, 5) to (4, 6):

- …, (1, 5), (1, 6), (2, 6), (3, 6), (4, 6), …
- …, (1, 5), (2, 5), (2, 6), (3, 6), (4, 6), …
- …, (1, 5), (2, 5), (3, 5), (3, 6), (4, 6), …
- …, (1, 5), (2, 5), (3, 5), (4, 5), (4, 6), …

Meanwhile, there is only one unique path from (1, 1) to (1, 5) and from (4, 6) to (4, 10).

Input

The first line of input contains an integer T ($T \leq 100$) denoting the number of cases. Each case begins with three integers: N, M, and K ($2 \leq N, M \leq 1{,}000{,}000$; $0 \leq K \leq 10$) denoting the size of the map and the number of punching robots respectively. The following K lines, each contains two integers: R_i and C_i ($1 < R_i < N$; $1 < C_i < M$) denoting the position of i[th] robot (row and column respectively) in the map. You are guaranteed that, for any two robots, the row difference or the column difference will be at least 3, i.e. no two robots' punching areas are overlapping. You are also guaranteed that cell (1, 1) and cell (N, M) are not in punching areas of any robots.

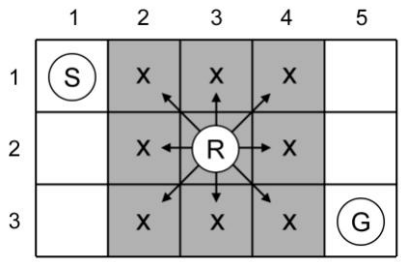
Output

For each case, output "`Case #X: Y`", where `X` is the case number starts from 1 and `Y` is the answer for that case modulo by 997.

Sample Input	Output for Sample Input
4 4 10 2 3 3 2 8 3 5 1 2 3 5 5 0 10 9 3 9 3 6 8 3 4	Case #1: 4 Case #2: 0 Case #3: 70 Case #4: 648

Explanation for 2[nd] sample case

The following figure represents the map for the 2[nd] sample case.



As you can see, there is no way you can reach (3, 5) from (1, 1) without being punched by the robot.