ACM ICPC Asia Kuala Lumpur Regional Contest 2014

31 October 2014 – 1 November 2014

Problem Set 2014

This problem set contains 12 questions (A-L) in 26 pages

ACM ICPC Malaysia Office

Kulliyyah of Information and Communication Technology
International Islamic University Malaysia
Kuala Lumpur, Malaysia

acm International Collegiate Programming Contest

IBM event sponsor

A	THE MOUNTAIN OF GOLD?	
	Input	Standard Input
	Output	Standard Output
	Time Limit	1 second

Problem Description

Ancient history and myth points to the Gunung Ledang (a.k.a. Ledang Mountain, as "gunung" is mountain in Malaysian) being the site of rich gold deposits, luring traders from as far as Greece and China. In the 14th Century, the Chinese seafarers plying the Straits of Melaka called it 'Kim Sua' meaning the 'Golden Mountain'. The mountain was named 'Gunung Ledang', which means 'mount from afar', during the period of the Majapahit empire. Legend has it, that before the death of Princess Gunung Ledang, she hid a huge amount of gold far back in time during the creation of earth in Ledang Mountain. It is (apparently, becoming "was") a mystery as to how the princess was able to go so far back in time. However, the princess was known to possess mystical powers that could enchant any pool she bath in into a portal that could manipulate time and space. A Malaysian historian of this time have discovered many of these pools located near many mountains around the world and named them "Ledang Pools". A Ledang Pool is a portal through space and time connecting two pools. Ledang Pools have a few peculiar properties:

- A Ledang Pool is a one way portal with two end points, i.e. it connects exactly two mountains.
- The time it takes to travel through a Ledang Pool is negligible.
- Each mountain might have multiple Ledang Pools' end point in its area.
- For some unknown reasons, starting from Ledang Mountain in Malaysia, it is always possible to end up in any mountain (of course, which also has Ledang Pool end point) on earth by hopping a sequence of Ledang Pools.
- There are no Ledang Pools with both end points on the same mountain area.
- Each Ledang Pool has a fixed time difference (distortion) between their end points. For example, traveling through a certain Ledang Pool may cause that person to end up 42 years in the past at the other end point. It's a space and time traveling!

The Malaysian historian suspects that a large amount of gold is hidden in Ledang Mountain in the past because there's no gold found in this mountain at this time (but then, where does

the myth come from?). Starting from Ledang Mountain, the historian wants to reach the same mountain (Ledang Mountain) in the past. In order to this, he has to hop into two or more Ledang Pools, doing the space-and-time-travel things, and end up at Ledang Mountain in the past. Note that he does not care in which time in the past he will end up at Ledang Mountain as long as it is in the past. Your task is to help this historian to determine whether it is possible to reach Ledang Mountain in the past from Ledang Mountain in the present time by hopping a sequence of Ledang Pools.

Input

The input file starts with a line containing the number of cases T ($1 \leq T \leq 20$). Each case starts with two numbers N and M in a line. These indicate the number of mountains which have Ledang Pool end point ($1 \leq N \leq 1{,}000$) and the number of Ledang Pools ($0 \leq M \leq 2{,}000$). The pools are numbered from 0 (Ledang Mountain in Malaysia) up to N-1. The next M lines each contains three integers A, B, C ($0 \leq A, B < N$; $A \neq B$; $-1{,}000 \leq C \leq 1{,}000$) denoting the properties of one Ledang Pool, i.e. someone can do a space-time-travel from mountain A to mountain B and he will end up C year in the future/past. If C is positive, then it is in the future, otherwise it is in the past.

Output

For each case, output in a line "Case #X: Y" (without quotes) where X is the case number starting from 1, and Y is “possible” (without quotes) if it is indeed possible to reach Ledang Mountain in the past, otherwise output Y as “not possible” (without quotes).

Sample Input	Sample output
2 2 2 0 1 15 1 0 -20 4 4 0 1 10 1 2 20 2 3 30 3 0 -60	Case #1: possible Case #2: not possible

B	SEQUENCE	
	Input	Standard Input
	Output	Standard Output
	Time Limit	1 second

Problem Description

Mr. Adam loves to solve challenge! He is given a binary sequence. For K times, Mr. Adam should choose an element of the sequence, and flip it (changes from 0 to 1 or 1 to 0, depends on the element). In this challenge, the final steps of the sequence should be a sequence contains only zeroes.

Mr. Adam can solve this challenge easily. However, he starts to wonder in how many ways he can solve this challenge. Now, it becomes your challenge! For this problem, you only need to compute that number modulo by 1,000,000,007 ($10^9 + 7$).

Note: Two ways are considered to be different if there is an i such that at i-th time, Mr. Adam chooses different elements.

Input

The first line of input contains an integer T ($1 \leq T \leq 50$), the number of cases. The next lines describe T cases.

Each cases starts with a line consists of two integers N ($1 \leq N \leq 1,000$) and K ($1 \leq K \leq 1,000$) separated by a space, denotes the length of the sequence and the same K as described in the description. The next N lines, each consists of either 0 or 1, which is an element of the binary sequence. The integers in the input are given in the same order as the sequence.

Output

For each case, output in a line "Case #X: Y" (without quotes) where X is the case number starting from 1, and Y is the number of ways to solve this challenge modulo by 1,000,000,007 ($10^9 + 7$).

Sample Input	Sample Output
5 1 10 0 1 11 1 1 10 1 2 30 0 0 3 10 0 0 0	Case #1: 1 Case #2: 1 Case #3: 0 Case #4: 536870912 Case #5: 14763

C	TURTLE GRAPHICS	
	Input	Standard Input
	Output	Standard Output
	Time Limit	1 second

Problem Description

Turtle graphics is a popular way for introducing programming to kids. It was part of the original Logo programming language developed by Wally Feurzig and Seymour Papert in 1966. It involved the Turtle, originally a robotic creature that moved around on the floor. It can be directed by typing commands into the computer. The command ‘F’ (forward) causes the turtle to move forward one step, ‘R’ (right) rotates the turtle 90 degrees clockwise while leaving it in the same place on the floor, and ‘L’ (left) rotates the turtle 90 degree anticlockwise and leaving it in the same location.

When the turtle moves forward it will leave a trail. We will represent the trail with character 'x'.

For example, the command

FFFRFFFFLFFFRFFFFF

will create the pattern below

```
y 0.............................
    9.............................
    8.........xxxxxx..............
    7.........x....................
    6.........x....................
    5.....xxxxx....................
    4.....x........................
    3.....x........................
    2.....o........................
    1.............................
    0.............................
     0123456789012345678901234567890  x
```

with assumption that the initial direction of the turtle is facing to the North (the y-axis) and the initial location of the turtle is marked as 'o'. The maximum size of the drawing area is 64x64 with the origin coordinate (0, 0) at the lower left corner. You may safely assume that the path of the turtle never exceed the drawing area.

Input

First line of the input contains T the number of test cases ($1 \leq T \leq 50$). Each test case consists of two lines. The first line of each test case contains a pair of integer, x and y ($0 \leq x, y < 64$) which represents the starting point of the path. The second line contains a string with no more than 128 characters. Each character of the string is either 'F', 'R' or 'L' which represents a computer command for the turtle in order.

Output

For each case, output in a line "Case #X: x y n" (without quotes) where X is the case number starting from 1. x and y represents the final location of the turtle and n is the number of locations in which the turtle visits more than once.

Sample Input	Sample Output
2 5 2 FFFRFFFFLFFFRFFFFF 2 2 FFFRFFFRFFRFFFF	Case #1: 14 8 0 Case #2: 1 3 1

D	CIRCLE AND MARBLE	
	Input	Standard Input
	Output	Standard Output
	Time Limit	1 second

Problem Description

Adi has fallen in love with Putri and they have been in relationship for years. Adi is now ready to propose Putri and ask her to marry him. However, Putri doesn't seem want to make things easy for Adi. She asked Adi to play a game with her, and if he can win against her, then she will marry him.

At first, Putri drew one circle and put some marbles inside it. Next, she drew another circle and put some marbles inside it. She also drew one arrow from the previous drawn circle which point to this new circle. After that, she drew another circle, put some marbles inside it, and drew one arrow from one of the previously drawn circle to this new circle. These steps are repeated until she drew N circles, each with marbles (some circles might be empty). Note that no circles intersect each others and no circle contains another circle. After she had drawn those N circles, Putri then said "Let's play a game".

"We alternately take turn in this game. In each turn, the player should choose one circle … let say it's the chosen circle. Take exactly one marble from the chosen circle, and move that marble to one of the circles which is pointed by the arrow originated from the chosen circle," she explained. "The one who cannot make his or her move, lose", she added. Wondering about this game, Adi then asked, "How about those circles which do not have any arrow originated from them? Can we take marble from those circles?" Putri then replied, "Ahh.. no, you cannot choose those circles. It's a mandatory that you move one marble to another circle. Since you cannot move any marble from those circles, then you cannot choose them." Putri then added, "Remember, you should move exactly one marble in your move, so obviously you cannot choose circles which do not have any marble inside it. Oh, by the way, the rule also applies to me".

Adi realized that this kind of game has a fool-proof strategy. It means if both players play optimally, then the outcome of the game depends only on the initial configuration. Adi then asked Putri one crucial information, "So… who moves first?". Putri replied, "I'll let you decide that". Adi then put a big smile on his face.

Given the initial game configuration, determine whether Adi should be the first player or the second player to be able to win the game. Assume Putri play optimally; in other words, Putri will surely beat Adi whenever she sees the chance.

Input

The first line of input contains an integer T ($T \leq 100$) denoting the number of cases. Each case begins with an integer N ($3 \leq N \leq 20{,}000$) representing the number of circles drawn by Putri. The circles are numbered from 1 to N. The next line contains N integers M_i ($0 \leq M_i \leq 1{,}000{,}000$) denoting the number of marbles in ith circle respectively ($1 \leq i \leq N$). The following line contains N integers P_i ($1 \leq P_i < i \leq N$) denoting which circle has an arrow pointing to it. P_1 is always 0, as it is the first circle drawn by Putri. This means no circle points to circle 1.

Output

For each case, output in a line "Case #X: Y" (without quotes) where X is the case number starting from 1, followed by a single space, and Y is “first” (without quotes) if Adi should take the first move in order to win the game, or “second” (without quotes) if he should play as second player.

Sample Input	Sample Output
4 3 1 1 2 0 1 2 3 1 2 2 0 1 2 4 1 1 2 3 0 1 1 3 5 3 2 0 1 4 0 1 2 2 4	Case #1: first Case #2: second Case #3: first Case #4: first

For ease of explanation, let's define some notations:

- move(a, b) as moving one marble from circle a to circle b.
- $\langle m_0, m_1, m_2, \ldots, \rangle$ as the number of marbles in circle 1, 2, 3, …, respectively.

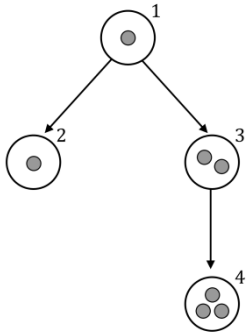
Explanation for 1st sample input
First player plays move(2, 3) resulting ⟨1, 0, 3⟩. Second player has no choice but to response with move(1, 2) resulting ⟨0, 1, 3⟩. First player win the game with move(2, 3) resulting ⟨0, 0, 4⟩.

Explanation for 2nd sample input
No matter what the first player does, he/she cannot win this game. There are 2 moves that can be played by the first player:

- move(1, 2), resulting ⟨0, 3, 2⟩; the two players then take turn moving marbles on circle 2 to circle 3, and this game will be won by the second player.
- move(2, 3), resulting ⟨1, 1, 3⟩; the second player then counter with move(2, 3) leaving ⟨1, 0, 4⟩. The first player has no choice but to play move(1, 2), which continued by the second player with move(2, 3) to conclude the game.

Therefore, in this game, Adi should be the second player in order to win.

Explanation for 3rd sample input
The following figure corresponds to the game configuration.



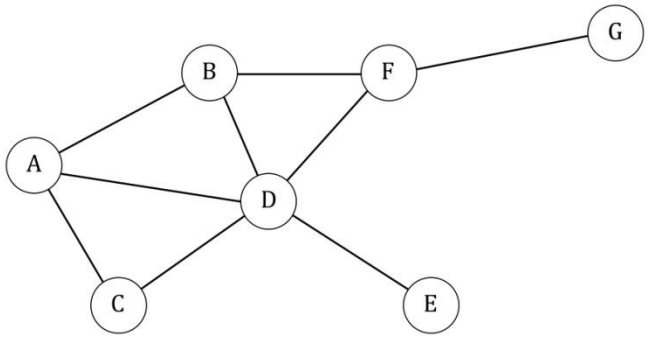
First player plays move(1, 2) resulting ⟨0, 2, 2, 3⟩. Second player has no choice but to response with move(3, 4) resulting ⟨0, 2, 1, 4⟩. First player then move the only marble in circle 3 to circle 4 and win the game.

E	GROUP OF STRANGERS	
	Input	Standard Input
	Output	Standard Output
	Time Limit	2 seconds

Problem Description

Online social networking services such as Facebook, Google+, Instagram, etc. has grown rapidly in the last decade. Nowadays, almost all youngsters, especially those on developed countries, know and use social networking services in their daily life. In this service, user creates account to represent him/her. Depends on the type of the services, each user establishes connections to other users in the same social network service. For example, in Facebook, this connection is called "friend" and it is bidirectional; it means if A is a friend to B, then B is also a friend to A. Meanwhile, in services such as Google+, any connection does not necessary bidirectional. You can "follow" (create connection to) some other users without requiring them to follow you back. In this problem, we regard each connection as bidirectional.

Supposed you have relationship information of N users in a certain social networking services. In how many ways can you choose three users such that those three are strangers to each other? In other words, these three persons do not have any connection to each other. For example, consider relationship information of 7 users as shown in the following figure.



In this example, there are 7 ways to choose three users which are stranger to each other, namely: A-E-F, A-E-G, B-C-E, B-C-G, B-E-G, C-E-F, and C-E-G. If you observe it carefully, user D cannot exists in any of those group-of-three as he has connections to all other users except G, thus any group-of-three will contain at least one user whom D connects to. An invalid example would be A-B-E. Even though A and B does not have any connection to E, but A and B are connected to each other.

Note that two groups are considered different if and only if both groups differ by at least one member.

Input

The first line of input contains an integer T ($T \leq 60$) denoting the number of cases. Each case begins with two integers N ($3 \leq N \leq 5{,}000$) and M ($0 \leq M \leq 20{,}000$) denoting the number of users and connections in the given social networking service. The following M lines each contains two integers A and B ($1 \leq A, B \leq N$; $A \neq B$) representing a connection between user A and user B. You may assume that all connections are unique.

Output

For each case, output in a line "Case #X: Y" (without quotes) where X is the case number starting from 1, followed by a single space, and Y is in how many ways you can choose three users such that they are stranger to each other.

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

F	PANTUN GRADER	
	Input	Standard Input
	Output	Standard Output
	Time Limit	1 second

Problem Description

A pantun is a traditional form of oral Malay verse. It is believed to have evolved to its most current form in the 15th century, as evidenced by Malay manuscripts. It has been adapted by both French and British writers since the late 19th century. Victor Hugo (picture) is sometimes credited with its introduction to the Western world, where it is called the "pantoum." In its most basic form the pantun consists of a quatrain which employs an *abab* rhyme scheme. A pantun is traditionally recited according to a fixed rhythm and as a rule of thumb, in order not to deviate from the rhythm, every line should contain between 8 and 12 syllables (a.k.a. syllable count requirement). A pantun is a four-lined verse consisting of alternating, roughly rhyming lines. The first and second lines sometimes appear completely disconnected in meaning from the third and fourth, but there is almost invariably a link of some sort.

Most Malay words has the following syllable structure: 6 and above length of word has 3 syllables, 4 and 5 length of word has 2 syllables, and 3 and less length of word has 1 syllable. However there are two general exceptions: for 6 length of word that contain '*ng*' or '*ny*' within the word, it has 2 syllables. For 3 length of word that starts with vowel alphabet, it has 2 syllables. For example: "aku" has 2 syllables, "kau" has 1 syllable, "berlari" has 3 syllables, "belang" has 2 syllables, "si" has 1 syllable and "seorang" has 3 syllables.

Pantun rhythm has the following structure. The last two alphabets of the last word in alternating verses should be the same, i.e. 1st - 3rd verses, and 2nd - 4th verses. For example, consider the following pantun and observe the bold (and underlined) alphabets.

Dua tiga kucing berla**ri**
Mana nak sama si kucing bela**ng**
Dua tiga boleh kau ca**ri**
Mana nak sama aku seora**ng**

As you can see, the 1st and 3rd verses have a same rhythm (ended with "ri"), while the 2nd and 4th verses also have a same rhythm (ended with "ng"). This is an example of pantun with good rhythm (both pairs rhyme). If we analyze the syllable count of the above pantun, then:

First verse : 8 syllables (1+2+2+3)
Second verse : 10 syllables (2+1+2+1+2+2)
Third verse : 8 syllables (1+2+2+1+2)
Forth verse : 10 syllables (2+1+2+2+3)

This pantun also has a same number of syllables for each pair of alternating verse, which is also a sign of a good pantun.

As someone who loves pantun, there are so many pantun that you need to assessed. Thus, it is handy to have an automatic pantun early-evaluator (we call it early-evaluator as the result of this evaluator only serves to determine the priority of pantun which should be assessed first).We decided that the evaluator should score the pantun in the following way:

- 10 points for each verse which fulfill the syllable count requirement (maximum of 4 verses).
- 20 points for each pair of alternating verse which fulfill the rhythm requirement (maximum 2 pairs of alternating verses).
- 10 points for each pair of alternating verse which have the same number of syllables (maximum 2 pairs of alternating verses).

The given pantun might have an arbitrary number of verses, but you should only consider the first 4 verses at most. Penalty of 10 points are given for each extra verse. Your task is to build this pantun early-evaluator.

Input

First line of input is an integer N that represents the number of test case, followed by N ($1 \leq N \leq 50$) lines where each line of input contains several verses where each verse is separated by a symbol comma, and ended by a symbol period. The total number of alphabet in a pantun will be not more than 255 including symbol comma and period, and the minimum length of word is 2.

Output

For each test case, the output contains a line in the format "Case #X: A B C D E" where X is the case number (starting from 1), A is an integer represents the point for syllable structure marks, B is an integer represents the point for rhythm structure marks, C is an integer represents the point for having exactly same number of syllable, D is an integer represents the point for penalty and E represents the overall points (i.e. A + B + C – D).

Sample Input	Sample Output
3 Dua tiga kucing berlari, Mana nak sama si kucing belang, Dua tiga boleh kau cari, Mana nak sama aku seorang. Banyak udang banyak garam, Banyak orang banyak ragam. Aku kau, Dia saya, Aku kau, Dia saya, Aku kau, Dia saya.	Case #1: 40 40 20 0 100 Case #2: 20 0 0 0 20 Case #3: 0 40 20 20 40

G	HARI MERDEKA	
	Input	Standard Input
	Output	Standard Output
	Time Limit	3 seconds

Problem Description

Malaysia's Independence Day -- also known as Hari Merdeka -- is celebrated each year on 31 August to commemorate the independence of Federation of Malaya from British colonial rule in 1957. During the whole month of August, many Malaysians express their patriotisms and loves toward their country by raising Malaysian flag on their home's balcony and on any vehicles they have (baby stroller included). This independence day celebration is usually incomplete without shouting "Merdeka!" seven times.

For the next year Independence Day celebration, IIUM plans to put up a long banner around IIUM's main campus. The committees want to write meaningful and motivating words in this banner to inspire students (and possibly professors too). To make things interesting, they agree that all words should be written without spaces in one single line. Moreover, they also agree that words are allowed to overlap each other. For example, "WORDER" contains the words: "WORD" and "ORDER". For ease of explanation, let say whatever is written on the banner as text.

The committees have compiled a list of words and assigned each word with a score which will be counted as the text's score for every occurrence of such word in the text. For example, if the value of WORD is 5 and ORDER is 8, then the text WORDER has a score of 5 + 8 = 13; WORDWORDER has a score of 5 + 5 + 8 = 18 (notice that WORD occurs twice in this example).

Aside from words, the committees also want each character in the text to appear as fancy as possible, thus, each written character will be specially handcrafted. Of course this cause another problem: handcrafting each character requires additional cost and obviously the committees' budget is limited.

Your task in this problem is to help the committees to determine what text should be written on the banner such that its score is as high as possible while the cost of writing such text is within the given budget. To make things easier, you only need output the score.

Input

The first line contains an integer T ($T \leq 30$) denoting the number of cases. Each case begins with three integers in a single line: N ($1 < N \leq 26$), M ($1 \leq M \leq 100$) and B ($10 \leq B \leq 200$) denoting the number of characters, the number of words, and the committees' budget. The next N following lines each contain one character H_i (A-Z) and one integer C_i ($1 \leq C_i \leq 3$) which represent the cost of writing one character H_i in the text. You may assume H_i are unique for all i = 1..N. The following M lines each contain one word W_i ($1 \leq |W_i| \leq 100$) and one integer S_i ($1 \leq S_i \leq 100$) denoting the score of such word. Assume all characters in the given set of words exists among H_i.

Output

For each case, output "Case #X: Y" (without quotes) in a line where X is the case number, starting from 1, followed by a single space, and Y is the highest possible score which can be obtained in the corresponding case.

Sample Input	Sample Output
2 3 3 10 A 2 B 2 C 3 AA 10 AAA 30 ABC 60 14 8 200 M 2 I 3 R 2 A 1 O 2 E 2 C 2 L 2 F 1 V 2 Y 3 H 1 U 2 N 2 MIRACLE 100 FLUFFY 40 LOVE 100 IVY 10	Case #1: 130 Case #2: 2600

VEN 30 AYE 20 HOE 10 EURO 80	

Explanation for 1st sample input

The text with highest score is AAAAA. There are 4 occurrences of AA and 3 occurrences of AAA, which makes the score to be 4 * 10 + 3 * 30 = 130. Writing AAAAA requires 5 A each costs 2, thus the total cost is 10.

H	TÚNEL DE RATA	
	Input	Standard Input
	Output	Standard Output
	Time Limit	1 second

Problem Description

Arturo has just bought a huge mansion in the Andes Mountains in Chile to study up closely on the chinchilla, an endangered rat species. They live in underground burrows or tunnels deep under the mansion, and will run and chase there along their own burrow circuits. They only dig up new burrows after a 30 days cycle.

Arturo has commission you to install motion-sensitive cameras to automatically activate the recording along these tunnels when these chinchilla pass through them. There should be at least one motion-sensitive camera along each of the possible running routes as he wants to capture all possible routes and their habits using minimum number of cameras. The underground burrows network can be represented as a series of connected intersections and bidirectional tunnel lanes (see Figure H). A possible running route consists of a start intersection A followed by a path consisting of two or more lanes (B, D, or E) that eventually leads back to the start intersection A. These running routes can start from any intersection.

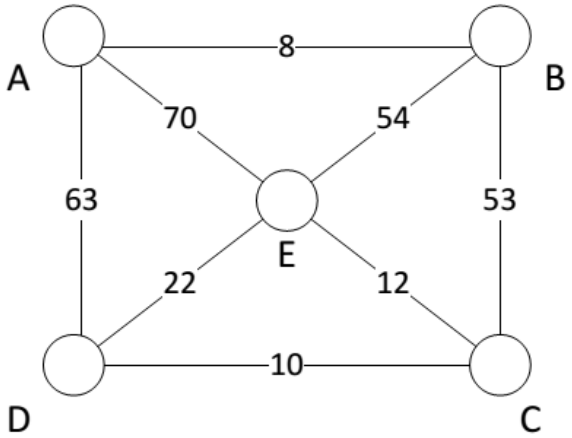


Figure H: A sample burrow of intersections and connecting bidirectional tunnel lanes

Each running route starts and ends at the same intersection, and each lane in that running route can be travelled once only. The cameras are deployed on the lanes (e.g. between C and D, etc.) and not at the intersections. The cost of deploying a camera depends on the length of

the lane on which it is placed (e.g. in Figure H, the length of lanes A-B, B-C and D-E are 8, 53 and 22 respectively). Thus, your task now can be summarized as to select a set of lanes that minimizes the total cost of deployment of these motion-sensitive cameras while ensuring that there is at least one camera along every possible running route or loop in the burrow network. Write a program that computes the optimal placement of these cameras in the connected burrows.

Input

The input consists of a line containing the number **T** of test cases, followed by **T** test cases, 0 < **T** < 1,001.

The first line of each dataset contains two positive integers, **S** and **L**, separated by a blank, which represent the number of intersections and number of lanes, respectively. You may assume that 0 < **S** < 10,001 and 0 < **L** < 100,001. Each of the **S** intersections is labeled from 1 to **S**. The following **L** lines of each dataset describe one lane per line. Each line consists of three positive integers which are the labels of two different intersections followed by the length of this lane. The length of each lane is between 1 and 5,000.

Output

For each case, output "Case #X: D M" (without quotes) in a line where X is the case number, starting from 1, followed by a single space, the integer D is the minimal total cost of setting up the chinchilla running and chasing monitoring system such that there is at least one motion-sensitive camera along every possible route, and integer M is the maximum length of the lane where a motion-sensitive camera is to be installed.

Sample Input	Sample Output
1 5 8 1 2 8 1 5 70 1 4 63 2 3 53 2 5 54 3 4 10 3 5 12 4 5 22	Case #1: 52 22

I	BEST POSITION	
	Input	Standard Input
	Output	Standard Output
	Time Limit	10 seconds

Problem Description

Farmer John wants to build a new farm on a large field. The field is represented as a grid of size R x C. Each cell in the field can be used to produce a type of food: either grains (G) or livestock (L). Below is an example of a field of size R = 5, C = 8:

```
  12345678
1 GLGGLGLG
2 GGLGGLGL
3 GGLLLGGG
4 LLGLLGLG
5 LGGGLGLL
```

Farmer John already have a set of design blueprints of the farm he wants to build. Each blueprint is represented as a grid of size H x W, where H <= R and W <= C. Each cell in the blueprint denotes the type of food John wants to produce: either grains (G) or livestock (L). For example, a blueprint of size H = 2, W = 3:

```
  123
1 GLL
2 LGG
```

Using this blueprint, Farmer John can build the actual farm on a certain position in the field. The farm position is represented by the position of its top-left corner. Suppose the farm is built at position (r, c) in the field. The farm must entirely built inside the field (i.e., $r + H - 1 \leq R$ and $c + W - 1 \leq C$). If the type of food in the cell of the field at position (r + i, c + j) matches the type of food in the cell of the blueprint at position (i+1, j+1) where $0 \leq i < H$, $0 \leq j < W$, then the food can be produced. Farmer John wants to pick the farm position in the field such that the farm produces the most number of foods (grains + livestock). If there are more than one possible position, he prefers the top-most position and if there are still more than one possible position, he prefers the left-most position. From the given field and blueprint examples above, the best position is to build the farm at position (1, 3), which is the position of the top-left corner of the farm in the field. As shown in bold:

```
  12345678
1 GL**GGL**GLG
2 GG**LGG**LGL
3 GGLLLGGG
4 LLGLLGLG
5 LGGGLGLL
```

By building the farm at position (1, 3) in the field, Farmer John can produce 5 foods: 3 grains and 2 livestock. That is, for the first row of the blueprint, 1 grain and 1 livestock can be produced and for the second row of the blueprint, 1 livestock and 2 grains can be produced. Note that building the farm at position (2, 5) and (3, 2) also produce the same number of foods, however Farmer Johns prefer the top-most and then the left-most position. Placing the farm at any other position in the field will produce less than 5 foods.

Input

There is only one field in the input. The first line contains two integers R and C where $0 < R, C \leq 500$, followed by R lines each contains C characters describing the field. The next line contains an integer B where $0 < B \leq 5$, which denotes the number of blueprints Farmer John has, followed by B blueprints specifications. Each blueprint starts with a line containing two integers H and W where $0 < H \leq R$ and $0 < W \leq C$, followed by H lines each contains W characters describing the blueprint.

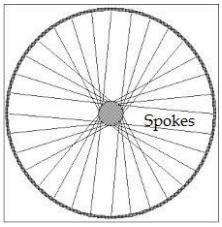
Output

For each case, output "Case #X: Y" (without quotes) in a line where X is the case number, starting from 1, followed by a single space, and Y is the four integers output separated by a space between them. The first two integers denote the best position to build the farm. The next two integers are the number of grains and livestock that can be produced.

Sample Input	Sample Output
5 8 GLGGLGLG GGLGGLGL GGLLLGGG LLGLLGLG LGGGLGLL 3 2 3 GLL LGG 3 1 L G G 1 4 GGLL	Case #1: 1 3 3 2 Case #2: 1 2 2 1 Case #3: 3 1 2 2

J	SPOKES WHEEL	
	Input	Standard Input
	Output	Standard Output
	Time Limit	1 second

Problem Description



Mr. Miyamoto is playing with a wheel. Each wheel can have many spokes. A spoke is one of the rods radiating from the center of a wheel (the hub where the axle connects), connecting the hub with the round traction surface. The wheel has 32 spokes with random of 0 or 1 as the value/tag for each spoke.

He has given a list of spokes, before and after its movement. His task is to calculate its optimal movement, whether the spokes move (in rotation) to the left or right. The list of spokes can be represented in a hexadecimal. For example: 800000001, means:

> 1000 0000 0000 0000 0000 0000 0000 0000 (list of spokes before its movement) and
> 0000 0000 0000 0000 0000 0000 0000 0001 (after its movement)

From this example, he found that there are two possible rotations: move to the left 1 (one) time and move to the right 31 times. The optimal rotation is the minimal number to move (in rotation), which is move to the left 1 (one) time.

Input

The first line of input gives the number of test cases, T ($1 \leq T \leq 1000$), followed by a line which content of the N1 as a state **before** the spokes move and N2 as a state **after** the spokes move. Whereas N1, N2 are 32-bits integer in hexadecimal representation (1..F)

Output

For each case, output in a line "Case #X: Y" (without quotes) where X is the case number starting from 1, and Y is the minimum number of wheel's spoke movements and followed by the direction to the "Left" or "Right". If the number of movements to the left or right is the same, the direction will be written as "Any". If N2 is not the final state of N1 then the output will be written as "Not possible".

Sample Input	Sample Output
4 80000000 1 1 80000000 AAAAAAAA 55555555 1 7	Case #1: 1 Left Case #2: 1 Right Case #3: 1 Any Case #4: Not possible

K	BALL	
	Input	Standard Input
	Output	Standard Output
	Time Limit	1 second

Problem Description

You are given some colored marbles arranged in a circle. The marbles can have three different colors: red, white and green. The marbles changes their color in each second. The new color of a marble depends on the marble right to it. The rules of changing color is as follows:

- if the color of the marble is white then in the next second its color will be the one on its right.
- otherwise, the marble does not change color if the color of the one of its right is white.
- otherwise, if the marble has the different color than the one of its right, it will become white.
- otherwise, the marble will have the same color with its right. In this case, if it is red then it will become green and if it is green it will become red.

You are given a string S and an integer N. Determine the state of the marbles after N seconds. If we consider S as a character array and its length is L, then we have L marbles in the circle. Marble (i+1) is in the right side of marble i. For marble (L-1), marble 0 is in its right.

Input

Input starts with T ($T \leq 20$), the number of test cases. Each of the next T line contains S ($1 \leq |S| \leq 20{,}000$) and N ($1 \leq N \leq 10^{18}$) separated by a single space.

Output

For each case, output "Case #X: W R G" (without quotes) in a line where X is the case number, starting from 1, followed by a single space, and W is the number of white marbles, R is the number of red marbles and G is the number of green marbles for that particular case, each separated by a single space.

Sample Input	Sample Output
5 RRGWRGW 2 WRGWRGWRGWRG 4 WGRWRGRRWGRWRG 3 RGRGWRGRG 3 RGRGRGRGWWWWRR 5	Case #1: 3 1 3 Case #2: 4 4 4 Case #3: 10 1 3 Case #4: 5 2 2 Case #5: 8 2 4

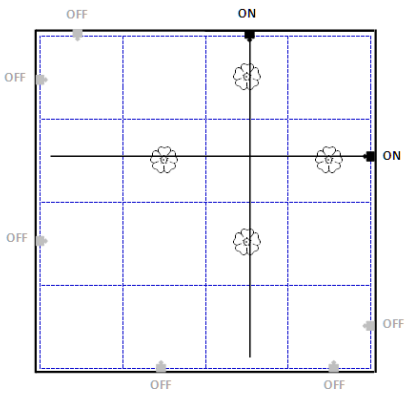
L	IRRIGATION LINES	
	Input	Standard Input
	Output	Standard Output
	Time Limit	1 second

Problem Description

A plantation consists of several rectangular fields, and a field is further subdivided into square zones. Under the multi-zone crop rotation, some zones are planted with crops, while some other zones are left fallow during a season. The zones of a field are rotated in this manner so that every few seasons, a zone would rest and be fallow. Crops of a season are selected based on their type, form, shape, and sun or shade requirements.

An irrigation system has been installed for each field. The system is constructed in a manner to simplify the operation and re-configuration of the system. The main water line runs around the field, which is depicted by the dark solid lines enclosing the field in the figure shown below. Dedicated control valves connect the mainline to the lateral (horizontal and vertical) irrigation lines that control the water flow to each of the rows and columns of zones. The valves are turned off if the irrigation line is not active. Only one lateral irrigation line is required to water a planting zone. The emitters or the holes on the active irrigation lines are closed over the zones that do not require to be watered. The layout of the planting field dictates the way the irrigation system is re-configured at the beginning of every season by turning off the valves of the inactive irrigation lines, and closing the unwanted emitters of the active lines.

To economise the management effort, the system needs to utilise a minimum number of lateral irrigation lines, i.e., lines that must be active. For example, the figure below shows a field divided into 4 x 4 zones. There are four zones that are planted with crops in the field marked by floral symbols, i.e., row 1 column 3, row 2 columns 2 and 4, and row 3 column 3. There are eight valves controlling the flow of water through their corresponding irrigation lines on the field. In the example, only the two lines whose valves have been turned on need to be activated to irrigate the crops on



the planting zones. These active irrigation lines are outlined by solid lateral lines in the figure. The inactive lines whose valves have been turned off are not drawn to avoid clustering of the figure.

Write a program that reads layouts of planting fields and determines the minimum number of lateral irrigation lines that must be activated in these fields.

Input

The first line contains an integer T ($T \leq 100$) denoting the number of cases. Each case describes a layout of a planting field, which starts on a new line with a pair of positive integers M and N ($1 \leq M, N \leq 100$), indicating the dimension of the field, i.e., the number of rows and columns of zones, respectively. The integers are separated by space. The next M lines of each test case delineate the layout of the field containing either a 1 or a 0, where 1 indicates the zone is planted and a 0 if it is fallow.

Output

For each case, output "Case #X: Y" (without quotes) in a line where X is the case number, starting from 1, followed by a single space, and Y is the minimum number of irrigation lines that must be active.

Sample Input	Sample Output
2 4 4 0010 0101 0010 0000 5 4 1001 0010 1100 1110 0101	Case #1: 2 Case #2: 4

--- end of the questions ---