Snake

Some day, Lea got overwhelmed with nostalgia. She rummaged through one of her old cabinets until she found her old mobile phone - a Nokia 3310. Then she sat in a corner for several hours to enjoy the simplicity of Snake.

Figure 1: Snake on a Nokia 3310[1]

Here, Lea controls a little snake that makes her way across a rectangular grid to munch on some little bits of food scattered around the playing field. If the head of the snake moves over a location with a bit of food, it grows in size by 1, i.e. the head moves into the location where the food is, but the tail does not move. If the snake moves outside of the grid, it reappears on the other side. Lea loses the game whenever the snake runs into some part of itself. However, the game seems to have changed a little from when she grew up (she asks herself what her mobile has been up to in that cabinet). Nowadays, all the little bits of food appear at the beginning of the game so Lea can now plan ahead where to go.

Can you tell her how her planned gameplay will work out?

Input

The first line of the input contains an integer t. t test cases follow, each of them separated by a blank line.

Each test case starts with a line containing two space-separated integers n and m, where n is the size of the playing field (an $n \times n$ grid) and m is the number of blocks of food bits. One line follows containing two integers c_s, r_s, the starting position of Leas snake. The snake starts in column c_s and row r_s with length 1 and always faces to the right. m lines follow, detailing the placement of the bits of food: The i-th line contains four space-separated integers c_i, r_i, w_i and h_i. This describes a rectangular block of food, with one bit of food at every grid location starting from column c_i and row r_i and extending w blocks to the right and h blocks down. If the starting point (c_s, r_s) is contained in this block, it will not contain food. A grid location can only ever contain one single bit of food, even if some of the m rectangular blocks overlap.

One line follows containing an integer l and a string s with l characters, describing Leas planned gameplay. The string

[1]Image source: https://apkpure.com/classic-snake-1997/com.byq.nokia_snake

contains one letter for every timestep with "F" being a step forward, "R" means the snake turns to the right, then steps forward and "L" means the snake turns to the left and then steps forward (all relative to the direction the snake is facing).

Output

For each test case, print a line containing "Case #i: $steps$ $points$" where i is its number, starting at 1, $steps$ is the amount of steps of Leas plan that can be executed successfully and $points$ is the amount of bits of food the snake will munch on until $steps$ steps have been reached. Each line of the output should end with a line break.

Constraints

- $1 \leq t \leq 20$
- $1 \leq n \leq 10^4$
- $0 \leq m \leq 100$
- $1 \leq l \leq 10^4$
- $1 \leq c_s, r_s \leq n$
- $1 \leq c_i \leq c_i + w_i - 1 \leq n$ for all $1 \leq i \leq m$
- $1 \leq r_i \leq r_i + h_i - 1 \leq n$ for all $1 \leq i \leq m$
- s contains only the letters "F", "R" and "L" and is of length l.

Sample explanation

For the first sample, in the first case, the grid is 4 by 4 with a bit of food in the upper left and lower right corner. Lea moves the snake to the right, then upwards for two steps, another step to the right (into the upper right corner) and then down to the lower right corner where the snake munches on the bit of food.

For the second case the grid is 10 by 10. The snake starts in $(1, 1)$ and makes its way to the lower right while picking up the two bits of food along the way.

In the third case the snake goes to the right, picking up all 6 bits of food, then turns around and hits itself at the 10-th step.

In the last case, the snake goes straight through the grid twice, picking up all 4 bits of food along the way, then turns around and hits itself. In this case the snake traverses several walls before it hits itself.

Sample Input 1

```
4
4 2
2 3
1 1 1 1
4 4 1 1
7 FLFRRFF

10 2
1 1
5 5 1 1
6 6 1 1
12 FFFFRFFFFLFF

10 1
4 5
5 5 6 1
13 FFFFFFLLFLRFF

5 1
1 2
1 2 5 1
15 FFFFFFFFFLLLRF
```

Sample Output 1

```
Case #1: 7 1
Case #2: 12 2
Case #3: 9 6
Case #4: 12 4
```

Sample Input 2

```
5
5 3
5 2
1 4 1 2
2 3 1 1
4 4 1 1
7 FFRFFFL

9 1
6 8
3 5 4 4
10 FFFFFFRRFF

10 3
6 1
9 8 2 1
9 1 2 1
1 1 1 1
9 RFFFLFFLL

4 1
1 1
1 1 4 4
8 LRFFFFLF

9 2
8 3
6 5 2 1
3 2 4 3
8 FFFFFFRF
```

Sample Output 2

```
Case #1: 7 1
Case #2: 10 1
Case #3: 9 0
Case #4: 4 4
Case #5: 8 4
```