Task 1

Your government has finally solved the problem of universal health care! Now everyone, rich or poor, will finally have access to the same level of medical care. Hurrah!

There's one minor complication. All of the country's hospitals have been condensed down into one location, which can only take care of one person at a time. But don't worry! There is also a plan in place for a fair, efficient computerized system to determine who will be admitted. You are in charge of programming this system.

Every citizen in the nation will be assigned a **unique** number, from 1 to P (where P is the current population). They will be put into a queue, with 1 in front of 2, 2 in front of 3, and so on. The hospital will process patients one by one, in order, from this queue. Once a citizen has been admitted, they will immediately move from the front of the queue to the back.

Of course, sometimes emergencies arise; if you've just been run over by a steamroller, you can't wait for half the country to get a routine checkup before you can be treated! So, for these (hopefully rare) occasions, an expedite command can be given to move one person to the front of the queue. Everyone else's relative order will remain unchanged.

Given the sequence of processing and expediting commands, output the order in which citizens will be admitted to the hospital.

Input

Input consists of at most ten test cases. Each test case starts with a line containing P, the population of your country ($1 \le P \le 1000000000$), and C, the number of commands to process ($1 \le C \le 1000$).

The next C lines each contain a command of the form 'N', indicating the next citizen is to be admitted, or 'E x', indicating that citizen x is to be expedited to the front of the queue. The last test case is followed by a line containing two zeros.

Output

For each test case print the serial of output. This is followed by one line of output for each 'N' command, indicating which citizen should be processed next. Look at the output for sample input for details.

Input	**Output**
`3 6`	`Case 1:`
`N`	`1`
`N`	`2`
`E 1`	`1`
`N`	`3`
`N`	`2`
`N`	
	`Case 2:`
`10 2`	`1`
`N`	`2`
`N`	
	`Case 3:`
`1000000000 11`	`9999`
`E 12`	`100000001`
`E 9999`	`100000001`
`N`	`13`
`E 100000001`	`10`
`N`	
`E 10`	
`E 13`	
`E 100000001`	
`N`	
`N`	
`N`	
`0 0`	

Task 2

Before bridges were common, ferries were used to transport cars across rivers. River ferries, unlike their larger cousins, run on a guide line and are powered by the river's current. Cars drive onto the ferry from one end, the ferry crosses the river, and the cars exit from the other end of the ferry.

There is an *l*-meter-long ferry that crosses the river. A car may arrive at either river bank to be transported by the ferry to the opposite bank. The ferry travels continuously back and forth between the banks so long as it is carrying a car or there is at least one car waiting at either bank. Whenever the ferry arrives at one of the banks, it unloads its cargo and loads up cars that are waiting to cross as long as they fit on its deck. The cars are loaded in the order of their arrival; ferry's deck accommodates only one lane of cars. The ferry is initially on the left bank where it broke and it took quite some time to fix it. In the meantime, lines of cars formed on both banks that await to cross the river.

The first line of input contains *c*, the number of test cases. Each test case begins with *l*, *m*. *m* lines follow describing the cars that arrive in this order to be transported. Each line gives the length of a car (in centimeters), and the bank at which the car arrives ("left" or "right").

For each test case, output one line giving the number of times the ferry has to cross the river in order to serve all waiting cars.

Input	**Output**
5	3
20 4	
380 left	3
720 left	
1340 right	5
1040 left	
	6
15 4	
380 left	6
720 left	
1340 right	
1040 left	
15 4	
380 left	
720 left	
1340 left	
1040 left	
15 4	
380 right	
720 right	
1340 right	
1040 right	
1 4	
100 right	
100 right	
100 right	
100 left	