Bounding Robots

As you design your new household helper robot, the SERVE-O-MATIC 1000, you are running it through a number of tests. In one test, you want to make sure that it always knows where it is. In particular, you're concerned about the situation where the robot runs into a wall that it is not aware of. For example, if it tries to walk forward 10 meters and runs into a wall it didn't know about after 3 meters, then there will be a 7-meter difference in where it stopped and where it *thinks* it stopped. Your robot does not yet have the intelligence to incorporate new information on the fly, such as this new wall, so it still thinks it walked 10 meters.

To test how bad this problem might be, you test a virtual robot in a virtual room, where the robot is not aware of the size of the room. The robot chooses a walking course, and your program simulates the robot's walk, keeping track of where it thinks it is, and where it actually is (by preventing it from walking through walls). After the simulation finishes, you want to know how far off the robot is from where it thinks it is.

Input

Input consists of up to 100 simulations.

Each simulation starts with a room description, which is two integers w and l (for width and length of the rectangular room, in meters). Each is in the range $[2, 100]$. The robot may walk anywhere in the rectangle marked by the corners $(0, 0)$ to $(w - 1,\ l - 1)$, including the edges (so it may walk to $(0, 0)$ or $(w - 1, 0)$, for example). The robot always starts at $(0, 0)$, and its steps are each exactly one meter long.

After the room description is a description of the path the robot plans to take. This description starts with an integer $1 \leq n \leq 100$, the number of walking segments in the path. This is followed by n segment descriptions to be followed in order. Each segment is given on a line as `x y`, where `x` is one of `u`, `d`, `l`, or `r` (for up, down, left, or right, respectively), and `y` is number of meters to move in that direction (in the range $[0, 30]$). Up and down move along the length of the room, left and right along the width. Up and right are in the positive direction, while down and left are in the negative direction.

Input ends when w and l are zero.

Output

For each simulation, output where the robot thinks it is as `Robot thinks x y`, and where it actually is as `Actually at x y` (for appropriate coordinate values x and y).

Sample Input 1

```
3 3
1
u 1
4 5
2
u 3
r 4
10 10
4
r 30
d 30
l 25
u 5
0 0
```

Sample Output 1

```
Robot thinks 0 1
Actually at 0 1

Robot thinks 4 3
Actually at 3 3

Robot thinks 5 -25
Actually at 0 5
```

Hide >

Please log in to submit a solution to this problem

Log in