Developer Programming Problem

Introduction

Don't stress about the challenge too much – it's supposed to be fun and a way for us to get a feel for your level of experience.

The problem is provided with sample data to be used for testing and the candidate should be able to demonstrate that their solution uses the supplied data in the form of a unit test or a simple user interface. User interface design is not the main focus of the problem.

Please complete this challenge using HTML, CSS and Javascript using the AngularJS framework.

Create a single page angular app, of any design you like or even no design at all. Create any input & output fields/areas/boxes/controls that you think you require. (e.g. one input field for the commands, and a scrolling area for the output, or alike)

Either zip your solution up and email it to us or put it on a public source repository (such as GitHub) and give us the URL. In either case, we should be able to run your code with no crazy steps.

Problem: Martian Robots

The Problem

The surface of Mars can be modelled by a rectangular grid around which robots are able to move according to instructions provided from Earth. You are to write a program that determines each sequence of robot positions and reports the final position of the robot.

A robot position consists of a grid coordinate (a pair of integers: x-coordinate followed by y-coordinate) and an orientation (N, S, E, W for north, south, east, and west).
A robot instruction is a string of the letters "L", "R", and "F" which represent, respectively, the instructions:

Left : the robot turns left 90 degrees and remains on the current grid point.

Right : the robot turns right 90 degrees and remains on the current grid point.

Forward : the robot moves forward one grid point in the direction of the current orientation and maintains the same orientation. The direction North corresponds to the direction from grid point (x, y) to grid point (x, y+1). There is also a possibility that additional command types maybe required in the future and provision should be made for this. Since the grid is rectangular and bounded (...yes Mars is a strange planet), a robot that moves “off” an edge of the grid is lost forever. However, lost robots leave a robot “scent” that prohibits future robots from dropping off the world at the same grid point. The scent is left at the last grid position the robot occupied before disappearing over the edge. An instruction to move “off” the world from a grid point from which a robot has been previously lost is simply ignored by the current robot.

The Input

The first line of input is the upper-right coordinates of the rectangular world, e.g. 5,3 for a world that is 6 wide, and 4 high (the lower-left coordinates are assumed to be 0, 0)

The remaining input consists of a sequence of each: 1 initial position and instructions two move, per robot (so two lines per robot) for as many robots as one wants to enter

A position consists of two integers specifying the initial coordinates of the robot and an orientation (N, S, E, W), all separated by whitespace on one line. A robot instruction is a string of the letters “L”, “R”, and “F” on one line.

So:

1,1E (instruction to position the robot on square 1,1 facing east)

RFR (instruction to tell the robot to turn right, go 1 forward, and turn right again)

You can enter as many of those 2 lines per robots as you like.

The Output

For each robot position/instruction in the input, the output should indicate the final grid position and orientation of the robot. If a robot falls off the edge of the grid the word “LOST” should be printed after the position and orientation.

Sample Input

```
5,3

1,1E
RFRFRFRF

3,2N
FRRFLLFFRRFLL

0,3W
LLFFFLFLFL
```

Sample Output

```
1,1E

3,3NLOST

2,3S
```