Leap Laboratories

Developer Programming Problem

Introduction

Think of this challenge as an opportunity to show us what “good” looks like to you; and a fun way to showcase your skills.

Here are some tips and guidelines:

- We don’t expect you to spend more than 2-3 hours on this challenge
- If you don’t have time to fully complete the challenge, please still send it in and indicate what your next steps would be. Remember to try to solve the hardest problems first.
- Use any language and frameworks you want
- Keep it simple
- User interface design is not the main focus of the problem
- Put your code on a public source repository (such as GitHub) and give us the URL
- Please submit your commit history, we are interested to see how you approach the challenge and how you verify the validity of your solution.
  We should be able to run your code without any crazy steps
- Imagine this is production quality code that you are submitting to your team for review

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 may be required in the future and provision should be made for this.

Since the grid is rectangular and bounded (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, the lower-left coordinates are assumed to be 0, 0.

The remaining input consists of a sequence of robot positions and instructions (two lines per robot). 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.

Each robot is processed sequentially, i.e., finishes executing the robot instructions before the next robot begins execution.

The maximum value for any coordinate is 50.

All instruction strings will be less than 100 characters in length.

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 1 E
RFRFRFRF

3 2 N
FRRFLLFFRRFLL

0 3 W
LLFFFLFLFL
```

Sample Output

```
1 1 E
3 3 N LOST
2 3 S
```