OBJECTIVE

The purpose of this take-home assignment is to enable you to demonstrate your proficiency in solving problems using software engineering tools and processes. Read the specification below and produce a solution. Your solution should be in the form of a completed and working Console application.

The problem specified below requires a solution that receives input, does some processing and

then returns some output. The UI should be a console application. Input should be read from Console standard in and output should be written to the Console via standard out. *There are multiple approaches to get to a working solution, so feel free to make assumptions whenever you find something ambiguous.*

The reviewer will be interested in:

* Your overall approach to this exercise and how you came to your solution. Please make sure to list your assumptions.
* Your ability to read and interpret the specification below.
* The architectural design and efficiency of your solution.
* The readability and supportability of your code.
* Your use of Unit Tests. At minimum, you should have tests proving that your solution works for the supplied sample input/output below.
* Professionalism of your code. Your code should follow standard C# coding styles, be free of major defects and be appropriately commented.

DELIVERABLES

Your solution must be written in **C#**. Create a zip archive containing all of your solution files. This assignment helps us better understand each candidate, plagiarism and cheating will not be tolerated and will result in immediate disqualification. As such, please make sure you Do NOT publish your solution to any public websites including github.

SPECIFICATION

A squad of robotic rovers are to be landed by NASA on a plateau on Mars. The navigation team needs a utility for them to simulate rover movements so they can develop a navigation plan.

A rover's position is represented by a combination of an x and y co-ordinates and a letter

representing one of the four cardinal compass points. The plateau is divided up into a grid to

simplify navigation. An example position might be 0, 0, N, which means the rover is in the bottom

left corner and facing North.

In order to control a rover, NASA sends a simple string of letters. The possible letters are:

* '**L**' – Make the rover spin 90 degrees **left** without moving from its current spot
* '**R**' - Make the rover spin 90 degrees **right** without moving from its current spot
* '**M**'. Move forward one grid point, and maintain the same heading.

Assume that the square directly North from (x, y) is (x, y+1).

INPUT

The first line of input is the upper-right coordinates of the plateau, the lower-left coordinates are

assumed to be 0,0.

The rest of the input is information pertaining to the rovers that have been deployed. Each rover

has two lines of input. The first line gives the rover's position, and the second line is a series of

instructions telling the rover how to explore the plateau.

The position is made up of two integers and a letter separated by spaces, corresponding to the x

and y co-ordinates and the rover's orientation.

Each rover will be finished sequentially, which means that the second rover won't start to move

until the first one has finished moving.

OUTPUT

The output for each rover should be its final co-ordinates and heading.

**Test Input:**

|  |  |
| --- | --- |
| **Input Type** | **Input Value** |
| Graph Upper Right Coordinate | 5 5 |
| Rover 1 Starting Position | 1 2 N |
| Rover 1 Movement Plan | LMLMLMLMM |
| Rover 2 Starting Position | 3 3 E |
| Rover 2 Movement Plan | MMRMMRMRRM |

**Expected Output:**

|  |  |
| --- | --- |
| Rover 1 Output | 1 3 N |
| Rover 2 Output | 5 1 E |