**Mars Rover Code Challenge**

A squad of robotic rovers are to be landed by NASA on a plateau on Mars.

This plateau, which is curiously rectangular, must be navigated by the rovers so that their on-board cameras can get a complete view of the surrounding terrain to send back to Earth.

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', 'R' and 'M'. 'L' and 'R' makes the rover spin 90 degrees left or right respectively, without moving from its current spot.

'M' means move forward one grid point, and maintain the same heading.

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

**Inputs**

**Input 1**: The first line of input is the size of the plateau (the upper-right coordinates). The lower-left coordinates are assumed to be 0,0.

```

Plateau Input: 8 4

```

**Input 2**: Landing co-ordinates for the Rover. 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.

```

Landing input: 1 2 N

```

**Input 3:** Navigation instructions (i.e a string containing 'L', 'R', or 'M').

```

Instructions Input: LMLMLMLMM

```

**Example**

Assuming 2 rovers have been created called "Rover1" and "Rover2".

**Input**:

```

Plateau: 5 5

Rover1 Landing: 1 2 N

Rover1 Instructions: LMLMLMLMM

Rover2 Landing: 3 3 E

Rover2 Instructions: MMRMMRMRRM

```

**Output**:

```

Rover1: 1 3 N

Rover2: 5 1 E

```

**Challenge**

Develop a command line app that can take the various inputs from the command line and generate the desired outputs.

**Expectations**

- app should be working and not crash

- bad input should be handled (doesn't have to be fancy)

- should be modular, readable, and configurable (not just one big function)

- should contain unit tests for critical functions

- should be coded in javascript (nodejs or in a browser)