**OOP design exercise #42.:**

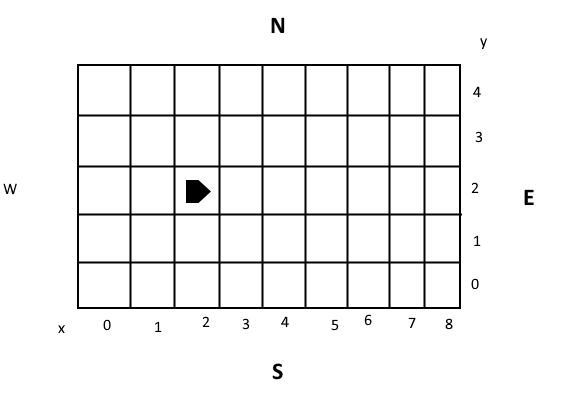
The purpose of this exercise is to evaluate your approach to software development covering among other things elements of Object Oriented Design, Software Design Patterns and Testing.  This exercise is not time bounded, we would prefer you take the time to design a good solution than rush something across to us.

Please complete the exercise in the language of your choice (preferable C#).  You are free to implement any mechanism for feeding input into your solution (for example, using hard coded data within a unit test). You should provide sufficient evidence that your solution is complete by, as a minimum, indicating that it works correctly against the supplied test data.

Please do not use any external libraries to solve this problem, but you may use external libraries or tools for building or testing purposes. Specifically, you may use unit testing libraries or build tools available for your chosen language (e.g., NUnit, Moq, etc.)

Imagine you have rectangular grid with 2 dimensions, length and depth. Each side of the grid represents cardinal directions of the world: North, South, East and West.

You have given pawn, which can be placed on the grid. This pawn has a frontside and can be moved ‘M’ to another field on the grid which is pointed by this frontside. You can also rotate the pawn 90 degrees, left ‘L’ or right ‘R’, without moving it from current spot. Assume that the square directly North from (x, y) is (x, y+1). In the picture below you can see pawn initial co-ordinates 2 2 E, grid dimension 8 4.



INPUT:

The first line of input is the grid dimension, the lower-left coordinates are assumed to be 0,0.

The rest of the input is information pertaining to the pawn. Each pawn has two lines of input. The first line gives the pawn’s position, and the second line is a series of instructions telling the pawn how to move on the grid.

The position is made up of two integers and a letter separated by spaces, corresponding to the x and y co-ordinates and the pawn’s orientation.

Each pawn will be finished sequentially, which means that the second pawn won’t start to move until the first one has finished moving.

OUTPUT

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

Example INPUT AND OUTPUT (not related to the picture)

Input Dimension - 5 5

Input starting point - 1 2 N

Input move sequence – LMLMLMLMM

Output end point – 1 3 N

Input Dimension - 5 5

Input starting point - 3 3 E

Input move sequence – MMRMMRMRRM

Output end point – 5 1 E

// My notes:

//

// The technical specification doesn’t specify any error cases (!!!)

// i.e. what should happen if the movement sequence is invalid,

// e.g. when the pawn reaches the edge of the board (grid) and is ordered to make more moves towards the edge. In such a scenario there appear to be 3 most obvious responses, that the system might be expected to follow:

1. The pawn pauses where it is and only continues to make moves when valid move sequences ensue.
2. The pawn’s position is “wrapped around” the board.
3. An error is indicated. The entire move sequence is reported back to the input provider as being invalid.

For the purpose of this exercise I will assume, that the system is expected to implement scenario resolution no 3.