**Pre-Interview Development Exercise**

The following is a practical programming test that should be completed by all prospective candidates prior to a face to face interview. Its function is to both give the candidate an opportunity to showcase their development skills, and to form a point of technical discussion in any forthcoming interview.

We expect all candidates will be able to submit the “correct” answers, so we are looking for coding standards and techniques that stand out above other candidates.

All candidates should submit a complete and well thought out solution, but are advised to pay particular attention to the expectations set out for each task to avoid taking the solution too far. We expect that a good candidate would spend no more than 2 hours.

Good Luck!

**Problem Specifications and Solution Expectations**

Complete the following task.

**Task – Robot Wars**

A fleet of hand built robots are due to engage in battle for the annual “Robot Wars” competition. Each robot will be placed within a rectangular battle arena and will navigate their way around the arena using a built-in computer system.

The arena is divided up into a grid to simplify navigation. A robot’s current location and heading is a combination two integers representing the x and y coordinates on the grid, and a single character value representing the current heading. The heading can be one of the four cardinal compass points (N, S, E and W). An example location might be 0, 0, N which means the robot is in the bottom left corner and facing North. The arena size is fixed at 5 by 5, giving 25 possible (x, y) coordinate positions. Assume that the grid point directly North-East from (x, y) is (x+1, y+1).

Each robot is controlled remotely by sending movement instructions. The simple communication protocol has the following two rules...

1. Only a single instruction can be sent at once, and
2. There are 3 valid instructions in the form of single alphabetic characters - ‘L’, ‘R’ and ‘M’.

‘L’ and ‘R’ make the robot spin 90 degrees to the left or right respectively without moving from its current grid point, while ‘M’ means move forward one grid point and maintain the same heading.

Each robot must be given an initial location at the start of the battle and must be able report its current location after any subsequent movements.

When a supplied movement instruction causes a robot to collide with the arena boundary then a penalty needs to be applied. The total number of penalties needs to be recorded so that bad navigation can be dealt with after the battle. If a collision occurs the position of the robot remains unchanged.

Your task is to implement the logic for a single robot to behave within the accepted parameters of the battle. For simplicity you can assume only a single robot is in the arena at once.

Code your solution in C# and show it works for the following scenarios...

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Initial Position** | **Movements instructions** | **Final position and penalty count** |
| **Scenario 1** | 0, 2, E | MLMRMMMRMMRR | Position: 4, 1, N Penalties: 0 |
| **Scenario 2** | 4, 4, S | LMLLMMLMMMRMM | Position: 0, 1, W Penalties: 1 |
| **Scenario 3** | 2, 2, W | MLMLMLM RMRMRMRM | Position: 2, 2, N Penalties: 0 |
| **Scenario 4** | 1, 3, N | MMLMMLMMMMM | Position: 0, 0, S Penalties: 3 |

Submission

* Submit a Visual Studio solution. Visual Studio Express edition can be downloaded for free and there are numerous free unit testing tools to choose from.

Pay particular attention to

* Showing the system can take the inputs and produce the outputs through a set of Unit Tests.
* Good code separation of concerns.
* Correct use of .Net Framework classes where appropriate.
* Clear and easy to understand C# code.

Hints

* There is no need to produce any UI component. Demonstrating the code works correctly with Unit Tests will be sufficient.