Prerequisites

===========

The Rover Application has been developed as a .NET Core 2 Console Application. The application may be run on Windows / Linux / macOS provided that the Core 2 runtime has been installed.

The .NET Core 2 runtime can be downloaded from <https://www.microsoft.com/net/download/windows>

Source Code

==========

The application source files can be found at

<https://github.com/FireAndIce68/Rover>

Running the Application

====================

The application can be run in the following ways:

* From Visual Studio

Open the Rover solution (Rover.sln) in Visual Studio 2017 and Run/Debug the *MoveRover* project. The project has been configured to run the *InstructionSet01.rcmd* command file which contains the commands given the specification.

* From the Command Line

The *Binaries* folder on GitHub directory contains compiled binaries. These can be copied to any local directory and the application executed from there. To run the *MoveRover* application open a command window / prompt and ensure that the current directory is the directory into which the files from the *Binaries* directory was copied. Also ensure that the *dotnet* command is accessible on the path. Type the following at the command prompt:

dotnet MoveRover.dll InstructionSet01.rcmd

The InstructionSet01.rcmd parameter may be replaced with any valid command file.

Solution Structure

===============

The solution consists of three projects:

* *RoverCore* which contains the classes that represent the Rover. The project consists of two primary classes
  + *CommandSet*: This class is responsible for parsing and validating and the commands received by the Rover. If valid the *CommandSet* is then executed by the Rover.
  + *Rover:* This class receives the commands, and if they parse into a valid *CommandSet* the *CommandSet* is first simulated and if the Rover will not move out of bounds the *CommandSet* is executed and the new position and orientation returned.
* *MoveRover* is a console application which sends instructions to and receives responses from the Rover and displays them to the screen.
* *RoversTests* contains the tests to validate the correct of the operation of the *RoverCore* code.

Specifications / Assumptions

=======================

The following assumptions were made / rules applied:

* The input format as given must be strictly followed. This implies the following:
* The input must consist of exactly three lines
* The first line must comprise of two and only two digits with no spaces between them. This implies the maximum grid is 99 or 81 square blocks.
* The second line must comprise two digits a space and one letter which must be one of N, E, S or W.
* The third line may be any length but must only contain the characters L, M or R.
* The Grid must be at least 2 square blocks.
* The entire instruction set must be well-formed or it will be ignored and the rover will not move.
* The Rover may not execute any instruction which will move it outside the grid. The instruction set will be pre-validated and if it would result in the Rover leaving the grid the entire instruction set will be ignored and the rover will not move.
* Instruction sets may be optimized to reduce movement - for example L followed R may be ignored or RRR may be replaced by L.
* The return string format was changed to be as follows N:X Y D:Message where N = Error count, X Y = new cartesian coordinates, D = new orientation and Message provides additional information.

Input

====

Input to the *MoveRover* application is done via a command file which consists of valid sets of command separated by at least one line starting the ‘#’ character.

# Command Set 1 – Optional first comment line Result 0:3 3 S

88

12 E

MMLMRMMRRMML

# Command Set 2 - Result 0:2 3 W

65

11 N

# Command Set 3 – Max Bound Error

MMRMRMMLLMML

88

12 E

MMMMMMMMMMMMMMMMMMMMM

Testing

======

The Test functionality of Visual Studio was used to build unit tests for the *RoverCore* primary classes. The tests ensure that:

* Valid commands produce a valid *CommandSet.*
* Malformed commands are detected and produce an invalid *CommandSet.*
* Well-formed but invalid commands (e.g. start position out of bounds) are detected and produce an invalid *CommandSet*.
* The Rover moves to the expected position when given a well-formed and valid commands.
* The Rover detects commands would cause it to move out of bounds.

The full set of tests can be run from Visual Studio 2017 by selecting Test | Run | All Tests or pressing Ctrl-R,A