Due end of day Thursday the 21st of March 2013 NLT 11:59PM

Lab Six CS0004 (CRN11118)

Beginning with this lab (and continuing over subsequent labs) we will implement a simple version of a popular board game. In our game two robots are trying to move from a start point (one corner of a square grid) to a destination point (a corner of square grid that is diagonally across from its start point). The robots begin on different corners that share an edge of the grid.

Using a limited hand of cards the players will command their robots to move forward or backwards in a straight line from 1 to 3 spaces, rotate left or right 90 degrees, or turn around. The robots will also fire a laser that may damage their opponent.

The locations on the square grid may move (or push) the robot, fire lasers at the robot, or rotate the robot.

Some squares have walls that block robot movement and some squares are pits that will kill a robot if they should move (or be moved) onto them.

If a robot moves onto a location where the other robot is; both will robots suffer a point of damage and the robot moving will push the other robot in the direction of movement. (If that robot cannot be pushed (prevented by a wall) then neither robot finishes movement but both still take damage.

In this game we will have two players (both human)

For this lab, the task is to create a project that contains a user interface (the form) that you think the program will have. You will set up the base comments and options as usual. Additionally, you should enter the shell outlines of the methods/procedures/functions that you think you will need.

As Examples:

If you think you will need a method called moveRobot and that it will need to know which Robot to move and which direction to move it. You might have the following code.

```
Private Sub moveRobot(ByVal player As Integer, ByVal grid(,) As String, ByVal direction As Integer)
End Sub
```

Or if you needed a function that returned whether a robot can move you might have the following code:

```
Private Function canMoveRobot(ByVal player As Integer,
                              ByVal grid(,) As String, ByVal direction As Integer) As Boolean
    Return True
End Function
```

Include comments in your code that describes what parts of the program you would implement first, second, and third and why you would implement them in that order. Also include a description of the various things you think you'll need to implement and how you would represent them.

For example: You might think that you will need to represent how much Health each robot has and that a two element integer array could do that with the first element being player 1's robot and the second element being player 2's robot. This might suggest that the integer value 0 could represent player 1 and the integer value 1 could represent player 2. Which might also suggest that who is first could be an integer value of 0 or 1 to indicate which player is currently the first player.

NOTE: This lab is about designing the solution before implementing it. So your comments should reflect your thoughts on how you would go about implementing this application; the general flow of the program and what/how the various objects will be represented.

Due end of day Thursday the 21st of March 2013 NLT 11:59PM

To begin the game each player draws a single card. The player with the highest speed value will begin. That player will choose a corner of the grid as the start position of her robot (meaning also that the destination will be the diagonally opposing corner.) The other player will choose a corner that is not the first players start location or the first players destination as is start position (the remaining unused corner is his destination.) Each Robot begins the game with 5 Health points.

A turn consists of:

- Each player gets 5 cards at random from deck. (If deck runs out of cards then the discard pile is shuffled and becomes the deck.
- Three rounds are played as follows:
  - Each player (in player order-determined by last card played) plays a card.
  - The player with the fastest speed (on card just played) becomes the 'first player' and her robot's position/orientation is changed based on the action described on the card she just played.
  - The other player's robot's position/orientation is now changed based on the card that player just played.
  - The locations that the robots are on will now perform any actions that they may do. The order of actions applied is pit, lasers, rotate, push, and move. If a robot is moved (or pushed) to a new location then only the non-movement actions of that new location will be applied. (The robot will not be moved or pushed twice by locations in a round.) A robot cannot move through a blocked direction. A robot cannot move into a square with another robot if that robot cannot be moved (pushed) in the same direction. If the robot can be pushed in that direction it will be pushed in that direction. (Note that blocking applies on exiting a location not on entering that location.)
  - Both robots now fire their lasers based on the cards played. Lasers travel until they strike an opponent, go off the edge of the grid, or are blocked when exiting a location (not when entering a location.)
    - *Damage to robots is applied to a robot as it occurs. If a laser hits robot it suffers 1 point of health for each laser that hits it. If a robot has fallen into a pit, its health is reduced to 0. If a robot pushed a robot (or is pushed by a robot or location) then it suffers 1 point of health for each time it pushes or is pushed). If a robot hits a wall either by movement or push it suffers a 1 point health loss.*
  - In player order, based on last card played, if any robot is below 0 Health then the game is over and the winner is the surviving robot.
  - In player order, based on last card played, if any robot has reached their destination then game is over and they win.
  - The card played is discarded.
- Players discard remaining two cards and a new turn begins.

Due end of day Thursday the 21st of March 2013 NLT 11:59PM

The table below lists all the cards in game. The first column specifies the action of that card and the remaining columns specify the speeds that will be found on a card of the action listed in first column of that row. There are 81 cards.

The program will read a file that contains this data. The name of the file is 'RoboRaceCards.txt' and the contents of the file will be a number of lines (corresponding to number of cards). Each line will be of the format: "Speed: <speed>, Action: <action>".

Using the table below our file will contain data like: (Only the first few lines of file are shown here.):

```
Speed: 450, Action: Backward 3
Speed: 360, Action: Forward 3
Speed: 010, Action: Turn Right 90
```
… etc.

Backward 3	450	530	600	660	710	750	780	800	810
Forward 3	360	440	520	590	650	700	740	770	790
Backward 2	280	350	430	510	580	640	690	730	760
Forward 2	210	270	340	420	500	570	630	680	720
Backward 1	150	200	260	330	410	490	560	620	670
Forward 1	100	140	190	250	320	400	480	550	610
Turn Around	60	90	130	180	240	310	390	470	540
Turn Left 90	30	50	80	120	170	230	300	380	460
Turn Right 90	10	20	40	70	110	160	220	290	370

The program will also read a file describing the grid the game will be played on. This file will be called 'RoboRaceMap.txt'

The first line will contain a number representing the size of the grid. (E.G. if the first line contained the number 11 then the grid will be an 11 by 11 grid.)

The remaining lines will contain data representing the nature of each location. (Using the example above; the next 11 lines will be the locations on the first row. Line one will be the first location of the first row, Line 2 will be the second location of the first row, … Line 12 will be the first location of the second row, and so on.

Each line will be of the format: "Action: <action(s)>"

The possible actions are:

- None
- Push North
- Push South
- Push East
- Push West
- Move North
- Move South
- Move East
- Move West
- Pit
- Block North
- Block South
- Block East
- Block West
- Rotate Left
- Rotate Right
- Laser

If a location has multiple actions then the actions are given as a comma separated list:

```
Action: Block North, Block East, Rotate Left, Laser
```