CS 1410 Introduction to Computer Science - CS2
Assignment #3
Total Points: 50 points

Problem description

In this assignment, you will be responsible for controlling a simulated robot around a generated board. Your program will parse a board from an input file. It will then move the robot around the board according to commands and quit. During game play, your program will also keep track of all items that were picked up by the robot.

The Board File

The game board will consist of the map dimensions, map layout, an initial robot position, and a list of commands. The first line of the file will be the map dimensions in the form “MAP N M” where N is the map height (row) and M is the map width (column). After that, there will be N lines containing M characters per line. Finally, the robot position will be indicated as “ROBOT X Y DIRECTION” where X is the column location, Y is the row location, and DIRECTION is either NORTH, NORTH-EAST, EAST, SOUTH-EAST, SOUTH, SOUTH-WEST, WEST, or NORTH-WEST. For example:

```
MAP 5 4
__D__
G____
___G_
D____
___D_
ROBOT 2 3 EAST
MOVE 1
PICKUP
MOVE -1
ROTATE 90
MOVE 1
PICKUP
MOVE -1
QUIT
```

This indicates a 5x4 map. In the 5 board lines, an underscore “_” represents a blank space, a D represents a diamond, and a G represents gold. The final line, “ROBOT 2 3 EAST”, indicates that the robot will begin at row 2, column 3 and face East. The rest of the commands will follow one per line and end with the QUIT command. Once the game stops, the results should be output to the screen

The Commands

During game play, the user will be allowed to enter several control commands. These commands include:

MOVE X This will move the robot X spaces in the appropriate direction. Negative will mean backwards.

ROTATE Y This will be the degrees, in 45 degree increments, that the robot should turn. 45.0 degrees indicates a clockwise and -45.0 a counter-clockwise rotation. Please note that this is a floating point value.

PICKUP This will indicate that the robot should pickup anything that is at his current location

QUIT This will stop the game and output the final results

The game logic should not allow the robot to leave the game board. If the command requests that a robot move to many spaces, it should move as many as possible and return an ERROR message. Also, do not forget to keep track of the number and types of picked up AND left behind items. Once the user has picked up all items on the board, the game should automatically output the results and quit. If an unrecognized command is found, an error should be reported to the screen and the invalid command skipped. If the move action tries to move the robot to far, the robot should be moved as far as possible, and an error reported. If the pickup command is executed on a space with nothing, then nothing should be picked up. If the rotate command is invalid, ie not an increment of 45.0, then it should be skipped.

You should create several sample files for testing your program. Try and use these files to fully test your program. The grader will use a specially designed test file to make sure everything is working.

Task 1 (5 points)

The first part of the assignment is to read a file name from the command line and open it. The file name will be the only argument specified. You man use any of the stringstreams for the file IO. *Remember to handle the case where the file argument is missing.*

Task 2 (15 points)

The second part of the assignment is to create the RobotGame class, process the map size, dynamically allocate room for the board, and process the board and robot location. The RobotGame class should house the board and all game related logic.

/* This constructor should allocate the 2 dimensional array into the private variable m_pBoard and initialize it.*/
[5 points] RobotGame(int rows, int columns);

/* This destructor should cleanup the board and any other dynamic memory */
[5 points] ~RobotGame();

/* This method is responsible for loading a single row of the board. It takes the row number and row c-string. This will be called for every row that needs parsing. */
[5 points] void loadRow(int row, char* pRowString);

Task 3 (30 points)

The third part of the assignment is to handle the commands. Remember, all commands are case insensitive. That means quit is the same as Quit is the same as QuIt is the same as...

/* This method is to output the gameboard and results to the screen. The game board should be 2 dimensional. */
[5 points] void DisplayGameBoard(); // Display the board

/* This function is responsible for moving the robot. It should move the robot as far as it can go and return the number of spaces moved. Remember, if the robot hits an edge of the board, it should stop (picture a robot hitting a wall). The return is so you can detect if an edge has been hit, ie if (moveRobot(5) < 5) it hit a wall.*/
[5 points] int moveRobot(int spaces);

/* This method is responsible for rotating the robot. It should only take degrees divisible by 45.0*/
[5 points] void rotateRobot(float degrees);

/* This method is responsible for running through all of the commands. It will be called by main after class construction and loadRow*/
[10 points] void play();

/* This method is responsible for handling the pickup command. It should keep track of the different types of items picked up*/
[5 points] void pickup();

Example Output

File:

MAP 5 4

__D__
G____
___G_
D____
___D_
ROBOT 2 2 EAST
MOVE 1
PICKUP
ROTATE 90
MOVE -1
PICKUP
JUMP 4
QUIT

Output:

Loading file input.txt
Loading board...
Board is:
```
    0  1  2  3  4
0  | _  _  D  _  _
1  | G  _  _  _  _
2  | _  _  _  G  _
3  | D  _  _  _  _
4  | _  _  _  D  _
```
Gold Left: 2, Gold Picked Up: 0, Diamonds Left: 3, Diamonds Picked Up: 0

Robot is facing East at position 2,3.
Board is:
```
    0  1  2  3  4
0  | _  _  D  _  _
1  | G  _  _  _  _
2  | _  _  R  G  _
3  | D  _  _  _  _
4  | _  _  _  D  _
```
Gold Left: 2, Gold Picked Up: 0, Diamonds Left: 3, Diamonds Picked Up: 0

Command Move forward 1 space to the East.
Board is:
```
    0  1  2  3  4
0  | _  _  D  _  _
1  | G  _  _  _  _
2  | _  _  _  R  _
3  | D  _  _  _  _
4  | _  _  _  D  _
```
Gold Left: 2, Gold Picked Up: 0, Diamonds Left: 3, Diamonds Picked Up: 0

Command Pickup at location 2,3.
Board is:
```
    0  1  2  3  4
0  | _  _  D  _  _
1  | G  _  _  _  _
2  | _  _  _  R  _
3  | D  _  _  _  _
4  | _  _  _  D  _
```
Gold Left: 1, Gold Picked Up: 1, Diamonds Left: 3, Diamonds Picked Up: 0

Command Rotate 90 from East. Now facing South.
Board is:
```
    0  1  2  3  4
0  | _  _  D  _  _
1  | G  _  _  _  _
2  | _  _  _  R  _
3  | D  _  _  _  _
4  | _  _  _  D  _
```
Gold Left: 1, Gold Picked Up: 1, Diamonds Left: 3, Diamonds Picked Up: 0

Command Move backward -1 space to the South.
Board is:
```
    0  1  2  3  4
0  | _  _  D  _  _
1  | G  _  _  R  _
2  | _  _  _  _  _
3  | D  _  _  _  _
4  | _  _  _  D  _
```
Gold Left: 1, Gold Picked Up: 1, Diamonds Left: 3, Diamonds Picked Up: 0

Command Pickup at location 1,3.
Nothings to pickup.
Board is:
```
    0  1  2  3  4
0  | _  _  D  _  _
1  | G  _  _  R  _
2  | _  _  _  _  _
3  | D  _  _  _  _
4  | _  _  _  D  _
```
Gold Left: 1, Gold Picked Up: 1, Diamonds Left: 3, Diamonds Picked Up: 0

Unknown command: JUMP
Board is:
```
    0  1  2  3  4
0  | _  _  D  _  _
1  | G  _  _  R  _
2  | _  _  _  _  _
3  | D  _  _  _  _
```

```
4 | _ _ _ D _
Gold Left: 1, Gold Picked Up: 1, Diamonds Left: 3, Damonds Picked Up: 0

Command is Quit.
You picked up 1 Gold and 0 Diamonds.
Thanks for playing
Board is:
    0 1 2 3 4
0 | _ _ D _ _
1 | G _ _ R _
2 | _ _ _ _ _
3 | D _ _ _ _
4 | _ _ _ D _
Gold Left: 1, Gold Picked Up: 1, Diamonds Left: 3, Diamonds Picked Up: 0
```

Extra Credit (10 points)

Implement an interactive mode. The interactive mode will a user to interact with the game. The map will still be loaded from a file. All of the commands must still be supported. This should not break your ability to perform all above options. The only changes in the file will be a modification to the map command, “INTERACTIVE_MAP X Y”, and a file devoid of individual commands. For example, a file might look like the following:

```
INTERACTIVE_MAP 5 4
__D__
G____
___G_
D____
___D_
ROBOT 2 2 EAST
```

Your program should be able to read from the file and decide which mode (interactive or scripted) to handle. If the mode is interactive, then the program must parse all commands from the user.