CSCI 2170 OLA 5 (Worth 125 points)
Due date: midnight Monday March 28th, Deadline: midnight Thursday March 31st

In this program you will complete a program to solve the maze problem. This program requires that you implement two classes: CreatureClass and MazeClass. The specification files of these two classes as well as the client program are given to you. You are required to complete the implementation files of the two classes according to the specifications given.

Instructions to download the program files and the data files:

Copy the following files from the directory **~cen/data/maze** into your project directory:

- **type.h** -- the file that defines the struct type "Coordinate" and the constants used in the program
- **CreatureClass.h** and **MazeClass.h** -- The specification files for the two classes
- **main.cpp** -- the client program
- **MyMaze1.dat**, **MyMaze2.dat,** and **MyMaze3.dat** -- the three data files.

The command to copy **all the files** in this directory at once is:

```
cp   ~cen/data/maze/*   codelite-workspace-directory/project-directory/.
```

The data file is formatted as the following:

```
20 7     ← size of the maze   number of columns  number of rows
0 18     ← the coordinates of the entrance location
6 12     ← the coordinates of the exit location
****************** *
*     *      ***** *
* ***** ***        *
* ***** *****   ** *
* *             *  *
* *******  *       *
************ *******
```

If a path exists from the entrance location to the exit location, it should found and the path is marked in the final maze output as shown below:

```
******************p*
*     *pppppp*****p*
* *****p***vppppppp*
* *****p*****   ** *
* *vvvvpppppp   *  *
* *******  *p      *
************p*******
```

If no path exists for the problem, a message is displayed.

Part A: Implement and test the MazeClass

<u>Create a new project OLA5A for this part of the project</u>

<u>MazeClass</u>

A MazeClass object consists of the following **data:**

- a maze (2D array of character),
- the number of rows in maze,
- the number of columns in maze,
- the entrance location of the maze, and
- the exit location of the maze;

and the following **methods**:

- **ReadMaze** that reads a maze from a file,
- **Display** that displays the maze,
- **GetEntrace** that returns the entrance location,
- **GetExit** that returns the exit location,
- **MarkVisited** that marks one maze location (parameter value passed in from the client program) as being visited,
- **MarkPath** that marks one maze location (parameter value passed in from the client program) as being part of the path being explored by a creature,
- **IsWall** that determines whether a location (parameter passed in from the client program) is wall,
- **IsClear** that determines whether a particular location (parameter passed in from the client program) is clear (of objects),
- **IsPath** that determines whether a particular location (parameter passed in from the client program) is part of the path being explored,
- **IsVisited** that determines whether a particular location (parameter passed in from the client program) has been visited and does not lead to exit,
- **IsExit** that determines whether a particular location corresponds to the exit location, and
- **IsInMaze** that determines whether a location provided as parameter passed in from the client program is part of the maze, i.e., is within the boundary of the maze.

Complete the implementation file (MazeClass.cpp) for the MazeClass, and then write a simple client program (ola5A.cpp) to test your MazeClass. This client program should:

- Create and initialize a maze object (including reading the maze from a data file),
- Display the maze
- Mark a location in the maze to be visited
- Mark a location in the maze to be path
- Display the maze again (check to see if the two locations above have 'v' and 'p' characters displayed)
- Display a message that reports the entrance and exit locations of the maze, and
- Display messages for 3 different locations of the maze whether it is wall, is clear, or is in maze. Do this in the steps below:
  - Assign a location corresponding to **wall** to variable "location"
  - Call the member function to test if this "location" IsWall? IsClear? IsInMaze? Display the result of the test.
  - Assign a location corresponding to **clear space inside the maze** to variable "location"
  - Call the member function to test if this "location" IsWall? IsClear? IsInMaze? Display the result of the test.

- Assign a location corresponding to **a space outside of the maze** to variable "location"
- Call the member function to test if this "location" IsWall? IsClear? IsInMaze?   Display the result of the test.

Part B: Implement and Test the CreatureClass

Create a new project OLA5B for this part of the project

CreatureClass

The only data a creature needs to keep up with as it moves around inside or outside the maze is its location expressed as a row-column pair, aka the coordinates.

- A creature can:
  - more up, move down, move left or move right, one step at a time.
- It can be assigned to a location, and
- It can report its current location.

Implement the CreatureClass according to the specifications given in CreatureClass.h. Write a simple client program (ola5B.cpp) to test this class. The client program should consist of the following steps:

- Create a creature object,
- Assign the creature to a location of your choice,
- Make the creature move x steps left, y steps down, z steps right, and q steps up, and then ask it to report its current location. (choose the values for x, y, z, q yourself).

After the MazeClass and the CreatureClass have been implemented and tested, move on to complete the maze problem:

- Create a new project named "OLA5"
- Create the two classes "MazeClasss" and "CreatureClass" in this project. You may copy and paste the content of the .h and .cpp files from the two classes from OLA5A and OLA5B projects.
- Copy and paste the content of the file "main.cpp" downloaded from ~cen/data
- Build and run the project using the 3 test data

What to turn in through D2L dropbox

You are required to submit these files:

- CreatureClass.h, CreatureClass.cpp
- MazeClass.h, MazeClass.cpp
- ola5A.cpp, ola5B.cpp
- main.cpp