CSCI 2170 OLA4
Part A (100 pts) electronic submission due midnight, Sunday, March 13th; hardcopy due beginning of class, Monday, March 14th

Part B (100 pts) electronic submission due due midnight, Sunday March 20th; hardcopy due beginning of class, Monday, March 21st

In this assignment, you will write a program to solve the maze problem using recursion plus backtracking. Detailed description of the program is given in the book (chapter 5).

The maze data file is formatted as the following:

```
7 20      ← size of the maze:  number of rows, number of columns
0 18      ← the coordinates of the entrance point (row index, column index)
6 12      ← the coordinates of the exit point (row index, column index)
****************** *
*      *       ***** *
* ***** ***          *
* ***** *****    ** *
* *                *  *
* *******  *         *
************ *******
```

Copy the 3 datafiles: MyMaze1.dat, MyMaze2.dat, and MyMaze3.dat, into your account with:
ranger$ **cp ~cen/data/MyMaze*.dat .** ← don't forget the trailing period

Program requirements:
(1) Prompt the user for data file to use
(2) Use dynamic memory allocation for the 2D array Maze
(3) Define two classes: **CreatureClass** and **MazeClass** (for MazeClass, you have to define your own destructor to free memory space dynamically allocated for the Maze).
(4) Implement recursive GoNorth(), GoSouth(), GoEast(), GoWest() client functions as described in the book.
(5) If a path exists between the entrance and exit points, print out the entire maze, including
    a. the path from the entrance point to the exit point marked by characters 'p',
    b. spaces explored by the creature by character 'v', as shown below:

```
******************p*
*     *pppppp*****p*
* *****p***vppppppp*
* *****p*****   ** *
* *vvvvppppppp   *  *
* *******  *p      *
************p*******
```

If no path exists for the problem, display an appropriate message and the entire maze after the exploration process.

Part A: (100 pts) Implement and test 2 classes: MazeClass and CreatureClass

MazeClass

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 location of the entrance location,
- **GetExit** that returns the location of the exit location,
- **MarkVisited** that marks one maze location being visited,
- **MarkPath** that marks one maze location is part of the path being explored by a creature,
- **IsWall** that determines whether a particular location in the maze is wall,
- **IsClear** that determines whether a particular location in the maze is clear of objects,
- **IsPath** that determines whether a particular location is part of the path being explored,
- **IsVisited** that determines whether a particular location has been visited and does not lead to exit,
- **IsExit** that determines whether a particular location in the maze is the exit point, and
- **IsInMaze** that determines whether a location provided is part of the maze.

You are required to use dynamic allocation of memory to create the 2D maze array. As a result, you need to explicitly define the destructor for the MazeClass to release memory space allocated when the maze object exits its scope.

Write the header file (MazeClass.h) and the implementation file (MazeClass.cpp) for the MazeClass, and then write a simple client program (ola4A1.cc) 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
- 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. Select the 3 locations such that one location is wall, one location is clear, and one location is outside of the maze.

CreatureClass

A creature can more up, down, left or right, one step at a time. It can be assigned to, or put at, a location, and can report its current location. The only data it needs to keep up with is its location, i.e., coordinates(x, y), at any given time.

First, write the header file (CreatureClass.h) and the implementation file (CreatureClass.cpp) of the CreatureClass. Then, write a simple client program (ola4A2.cc) to test this class. The client program should create a creature, assign it to a location of your choice, make it move x steps left, y steps down, z steps right, and q steps up, and then ask it to report its current location.

Instructions to submit your program

- **Hard copy**:
  - Create a script file by following the steps below:
    First, navigate to the directory where your program source file is located, then follow the steps below:

```
ranger$ script log4A                    ← this creates the log file named "log4A1"
ranger$ pr –n –t –e4 MazeClass.h
ranger$ pr –n –t –e4 MazeClass.cpp
ranger$ pr –n –t –e4 ola4A1.cc
ranger$ aCC MazeClass.cpp ola4A1.cc –o run4A1
ranger$ run4A1
ranger$ exit

ranger$ script log4A2                   ← this creates the log file named "log4A2"
```

```
ranger$ pr –n –t –e4 CreatureClass.h
ranger$ pr –n –t –e4 CreatureClass.cpp
ranger$ pr –n –t –e4 ola4A2.cc
ranger$ aCC CreatureClass.cpp ola4A2.cc –o run4A2
ranger$ run4A2
ranger$ exit
```

Enclose the hardcopy of the program and the program evaluation sheet in a folder

- **Soft copy:**
  - login the ranger system with www.cs.mtsu.edu/nx,
  - login to PeerSpace through the web browser provided by the ranger system, click on *tools|Assignments* to submit your softcopy.

Part B (100 pts)

Implement the client program (ola4.c) that solves the maze. Implement the recursive GoNorth(), GoSouth(), GoEast(), GoWest() functions as described in the text book and discussed in class. This program will include both the MazeClass and the CreatureClass header files.

Your program should be flexible – the entrance and exit points in a maze can be on any of the four sides.

Instructions to submit your program

- **Hard copy**:
  - Create a script file by following the steps below:
    First, navigate to the directory where your program source file is located, then follow the steps below:

```
ranger$ script log4
ranger$ pr –n –t –e4 CreatureClass.h
ranger$ pr –n –t –e4 CreatureClass.cpp
ranger$ pr –n –t –e4 MazeClass.h
ranger$ pr –n –t –e4 MazeClass.cpp
ranger$ pr –n –t –e4 ola4.cc
ranger$ aCC CreatureClass.cpp MazeClass.cpp ola4.cc –o run
ranger$ run          ←run 3 times with the three maze data files.
ranger$ run
ranger$ run
ranger$ exit
```

Enclose the hardcopy of the program and the program evaluation sheet in a folder

- **Soft copy:**
  - login the ranger system with www.cs.mtsu.edu/nx,
  - login to PeerSpace through the web browser provided by the ranger system, click on *tools|Assignments* to submit your softcopy.