Project 2

Specifications

1. Copy the folder Project2 from the course pickup folder.

2. Modify the makefile by replacing "yln" by your last name.

   P2.out: Proj2app.o CMaze.o
   g++ -g -o P2.out Proj2app.o CMaze.o

   Proj2app.o: Proj2app.cpp CMaze.h
   g++ -g -c Proj2app.cpp

   CMaze.o: CMaze.cpp CMaze.h
   g++ -g -c CMaze.cpp

   clean:
   rm *.o P2.out

   run : P2.out
   clear
   P2.out

   submit : Proj2app.cpp CMaze.h CMaze.cpp makefile README.txt
   rm -rf ylnP2
   mkdir ylnP2
   cp Proj2app.cpp ylnP2
   cp CMaze.h ylnP2
   cp CMaze.cpp ylnP2
   cp makefile ylnP2
   cp README.txt ylnP2
   tar cvfz ylnP2.tgz ylnP2
   cp ylnP2.tgz ~tiawatts/cs215drop

3. Document the file CMaze.h.

   // Add Documentation here

   #ifndef CMAZE_H
   #define CMAZE_H

   #endif // CMAZE_H
#include <iostream>
using namespace std;

enum sType {WALL, EMPTY, ME, GOAL, CRUMB};

class CMaze
{
    public:
    CMaze ();
    CMaze (const CMaze & other);
    ~CMaze ();
    CMaze & operator = (const CMaze & other);
    void Init (int R, int C);
    void Instructions (ostream & outs, istream & ins);
    void Display (ostream & outs);
    void Move (char direction);
    void Message (bool done, ostream & outs);
    bool Done ();
    bool finished;
    private:
    struct CMazeSquare
    {
    CMazeSquare ();
    void Display (ostream & outs);
    sType what;
    }
    void Temporary ();
    CMazeSquare ** grid;
    int numRows, numCols;
    int currentRow, currentCol;
    int endRow, endCol;
};

#endif


// Add Documentation here

#include <time.h>
#include "CMaze.h"

5. Document the function CMazeSquare::CMazeSquare.

CMaze::CMazeSquare::CMazeSquare ()
{
    // This function will:

    what = WALL;
6. Document the function CMazeSquare::Display.

7. If you wish, modify CMazeSquare::Display to enhance the user interface.

```cpp
void CMaze::CMazeSquare::Display (ostream & outs) {
    // This function will:
    char achar;
    switch (what) {
    case WALL:
        achar = 'X';
        break;
    case EMPTY:
        achar = ' ';
        break;
    case ME:
        achar = 'O';
        break;
    case GOAL:
        achar = '$';
        break;
    case CRUMB:
        achar = '.';
        break;
    }
    outs << achar << ' ';}
```

8. Document the function CMaze::CMaze.

```cpp
CMaze::CMaze () {
    // This function will:
    numRows = 1;
    numCols = 1;
    grid = new CMazeSquare * [1];
    grid[0] = new CMazeSquare [1];
    finished = false;
    currentRow = 0;
    currentCol = 0;
    endRow = 0;
    endCol = 0;
}
9. Document the function CMaze::~CMaze.

CMaze::~CMaze ()
{
    // This function will:
    for (int r = 0; r < numRows; r++)
        delete [] grid[r];
    delete [] grid;
}

10. Document the function CMaze::Init.

11. Replace the call to Temporary with C++ code to generate a maze using either the Recursive Division or Depth First techniques described in Homework 3.

void CMaze::Init (int R, int C)
{
    // This function will:
    numRows = R;
    numCols = C;
    grid = new CMazeSquare * [numRows];
    for (int r = 0; r < numRows; r++)
        grid[r] = new CMazeSquare [numCols];
    // Put your code for generating a maze here
    Temporary ();
    finished = false;
    currentRow = 1;
    currentCol = 1;
    grid [currentRow][currentCol].what = ME;
    endRow = numRows-2;
    endCol = numCols-2;
    grid [endRow][endCol].what = GOAL;
}

12. Document the function CMaze::Instructions.

13. Replace the strings in the output statement with appropriate instructions for your maze.

void CMaze::Instructions (ostream & outs, istream & ins)
{
    // This function will:
    outs << "Put your instructions here\n"
        << "Use backslash and n to create\nmultiple lines." << endl;
}
14. Document the function CMaze::Display.

```cpp
void CMaze::Display (ostream & outs)
{
  // This function will:

  int r, c;
  for (r = 0; r < numRows; r++)
  {
    for (c = 0; c < numCols; c++)
      grid [r][c].Display (outs);
    outs << endl;
  }
}
```

15. Document the function CMaze::Move.

16. Add code to each case in the switch statement to move the "ME" character from one CMazeSquare object to its neighbor as indicated by the letter entered by the user.

```cpp
void CMaze::Move (char direction)
{
  // This function will:

  switch (direction)
  {
    case 'a': case 'A':
      break;
    case 'w': case 'W':
      break;
    case 'd': case 'D':
      break;
    case 's': case 'S':
      break;
  }
}
```

17. Document the function CMaze::Message.

18. Replace the strings in the output statement with appropriate messages for your maze. This function will be called when the user successfully navigates the maze or when the user enters 'q' to quit the maze; the variable "done" will be true if the user has completed the maze and will be false if the user quits before finishing the maze.

```cpp
void CMaze::Message (bool done, ostream & outs)
```
// This function will:

outs << "Put your completion message here\n"
   << "Use backslash and n to create\nmultiple lines."
   << endl;
}

19. Document the function CMaze::Done.

20. Complete this function to return true if the 'ME' character has reached the "GOAL" character.

bool CMaze::Done ()
{
}

21. Replace the function "temporary" with the function (or functions) you have written to generate your maze.

// Replace this function with the function(s) used to generate your maze
void CMaze::Temporary ()
{
    for (int r = 1; r < numRows-1; r++)
        for (int c = 1; c < numCols-1; c++)
            if ((r % 2) || (c % 2))
                grid[r][c].what = EMPTY;
}

Date Due: 30 October 2016, 11:59 pm
To Turn In: YourLastNameP2.tgz containing a tarred and zipped version of a directory called YourLastNameP2. This should only contain the well documented files required to reconstruct your executable (P2.out). You must also include a file called README.txt in which you describe what works and what does not work in your submission.