

Project 3 – Part 1

For this project you will be implementing a Maze class and an application program that uses the Maze class. In this version of the project, the configuration of your maze will be fixed.

Specifications:

1. Create a file called Maze.h containing the following class specification:

```
#ifndef MAZE_H
#define MAZE_H

#include <iostream>
using namespace std;

class Maze
{
public:
    Maze ();
    Maze (const Maze & other);
    ~Maze ();
    Maze & operator = (const Maze & other);
    void Instructions (ostream & outs);
    void Display (ostream & outs);
    bool Move (char dir);
    void Message (bool completed, ostream & outs);
private:
    char ** grid;
    int numRows, numCols;
    int currentRow, currentCol;
    int endRow, endCol;
};

#endif
```

2. Create a file called Maze.cpp containing the implementations of the functions in the Maze class.

- a. The default constructor should allocate space for the maze grid and should initialize it to contain the maze you designed for the first preliminary exercise.
- b. The copy constructor and the assignment operator should appropriately create new copies of an existing maze.
- c. The destructor should release the space allocated to the maze grid.

- d. The Instructions function should output to the passed ostream object a set of rules describing the user's goal and interface for solving the maze.
 - e. The Display function should output to the passed stream object the current state of the maze.
 - f. The Move function will modify the maze to reflect a move in direction "dir". This function will also check to see if the end has been reached. If the end has been reached, the function returns true; otherwise it returns false.
 - g. The Message function should either congratulate the user or suggest that they play again as indicated by the passed Boolean flag. The message should be written to the passed ostream object.
3. Create an application file that instantiates an instance of the Maze class. It should call the Instruction, Display, Move, and Message functions for the Maze class object. A sample main function for your application program follows.

```
int main ()
{
    Maze M;
    M.Instructions (cout);
    M.Display (cout);
    bool solved = false;
    char d = ' ';
    while (!solved && d != 'q')
    {
        cout << "What direction would you like to move?";
        cin >> d;
        solved = M.Move (d);
        M.Display (cout);
    }
    M.Message (solved, cout);
    return 0;
}
```

4. Create a makefile for compiling your project. The resulting executable should be called "proj3".

Date Due: Tuesday, 4 November 2009 , 11:59 pm.

To Turn In: A tarred and zipped directory containing the above files. The directory should be called "yourlastnameP3" and the tar file should be called **yourlastnameP3.tgz**.

Project 3 – Part 2

For this part of the project you will be modifying the default constructor to “randomly” generate a maze using one of the 2 algorithms found on the wikipedia page:

http://en.wikipedia.org/wiki/Maze_generation_algorithm

1. Remove the fixed maze statements from your default constructor.
2. Create the prototype for a member function called Divide in the private section of your Maze class.
3. Implement the function Divide to generate the walls of your maze. This function should recursively call itself 4 times.
4. Use the srand (time(NULL)) command to start your random number generator. Use the rand () command to request “randomly” chosen integer values. You will need to include cstdlib and ctime.
5. Make sure that your makefile is generating an executable called “proj3”.

Date Due: Tuesday, 11 November 2009, 11:59 pm.

To Turn In: A tarred and zipped directory containing the above files. The directory should be called “yourlastnameP3” and the tar file should be called **yourlastnameP3.tgz**.