Department of Electrical and Computer Engineering
EECE 2560: Fundamentals of Engineering Algorithms

Project #5

In this project, you will develop algorithms that find paths through a maze.

The input is a text file containing a maze. Each maze begins with the number of rows and columns in the maze and a character for every cell in the maze. A cell contains a `O` if the solver is allowed to occupy the cell. A cell contains `X` if the solver is not allowed to occupy the cell.

The solver starts at cell `(0,0)` in the upper left, and the goal is to get to cell `(rows-1, cols-1)` in the lower right. A legal move from a cell is to move left, right, up, or down to an immediately adjacent cell that contains a space. Moving off any edge of the board is not allowed.

Part a

The maze is stored within the `maze` class. Functions to handle file I/O are included as part of the assignment.

In the printout of the graph, the current cell is represented by + and the goal cell is represented by *. These values are passed as parameters.

For each node, keep track of the maze cell `(i,j)` that it corresponds to. Use the `cell` property of a node to store this pair of values.

For each maze cell `(i,j)`, keep track of the node `v` that it corresponds to. Use a data structure in the maze class to store this information.

Add the following functions to the `maze` class:

```
void maze::mapMazeToGraph(Graph &g)
// Creates a graph g that represents the legal moves in the maze m.

void maze::printPath(Graph::vertex_descriptor end,
                     stack<Graph::vertex_descriptor> &s,
                     Graph g)
// Prints the path represented by the vertices in stack s.  Repeatedly
// calls print() to show each step of the path.

void clearVisited(Graph &g)
// Mark all nodes in g as not visited.

void setNodeWeights(Graph &g, int w)
// Set all node weights to w.

void clearMarked(Graph &g)
// Unmark all nodes.

ostream &operator<<(ostream &ostr, const Graph &g)
// Output operator for the Graph class.  Prints out all nodes and their
// properties, and all edges and their properties.
```

The code you submit should include a `main` function that initializes a maze by reading a file, converts the maze into a graph, and then prints out the graph.