COMP 2710: Lab 03 Design

**Analysis**

● Configure the Graph

The user will input the name of the file that holds the graph information beside the “Enter the name of the Maze configuration file:”. The program will take the .txt file and put it in node form.

● Find Shortest Path

The user will Configure the Graph (described above). The program will take the first line as the initialization spot and the second line as the destination spot. Then the program displays all the nodes visited, tell if the path was found, and (if found) list the shortest path.

● Quit

The user will enter “Quit” beside the “Enter the name of the Maze configuration file:”, and the program will display a thank you message and then stop running.

**Design**

MazeSolver

1

1

1

Map

1

1 - \*

1 - \*

Node

Node Class

VARIABLES (Private):

● string name

− name of the node (in this program, it is the map element value

● Node \*attachedNodes[4];

− this has the four possible linked node of the current node

● int numberUsed

− this keeps track of how many nodes are attached to the current node

● Node \*previous

− this is the previously linked node

● int marked

− this tells if the node is marked or not

FUNCTIONS (Public):

● Node (string newName)

− this is a constructor that creates the name of the node upon creation

● Node( )

− this is a constructor that creates a node without initializing the name of the node

● void setNodeName(string newName)

− this sets the value of the “name” variable

● string getNodeName( )

− this gets the value of the “name” variable

● void attachNewNode(Node \*newNode, int index)

− attach the inputted node at one of the four available indexes

● Node \*getAttachedNode(int index)

− this returns the attached node at the specific index

● int getNumberUsed( )

− this returns the value of the “numberUsed” variable

● int setNumberUsed(int newNumberUsed)

− this sets the value of the “numberUsed” variable

● void setPrevious(Node \*previousNode)

− this sets the value of the “\*previous” variable

● Node \*getPrevious( )

− this returns the value of the “\*previous” variable

● int setMarked( )

− this changes the node’s status from unmarked (0) to marked (1)

Map Class

VARIABLES (Private)

● Node[ ] map

− this is an array of nodes that makes up the current map

● Node initialNode

− this is the node that begins the path

● destinationNode

− this is the node that ends the path

FUNCTIONS (Public)

● void createMap(File textFileIN)

− this takes the inputted .txt file and sets the value of the “map”, “initialNode” (first line of .txt file), and “destinationNode” (second line of the .txt file) variables

● Node \*getMap( )

− this returns the value of the “map” variable

● Node getInitialNode( )

− this returns the value of the “initialNode”

● Node getDestinationNode( )

− this returns the value of the “destinationNode”

MazeSolver Class

VARIABLES (Private)

● Node initialNode

− this is the node that starts the path

● Node desitnationNode

− this is the ending node of the path

● Map currentMap

− this is the Map object representing the map from which the path is determined

FUNCTIONS (Public)

● void setInitialNode(string nodeNameIN)

− this sets the value of “initialNode” variable

● Node getInitialNode( )

− this returns the value of the “initialNode” variable

● void setDestinationNode(string nodeNameIN)

− this sets the value of the “destinationNode” variable

● Node getDestinationNode( )

− this returns the value of the “destinationNode” variable

|  |
| --- |
| 1 |
| Configure Graph |

User

|  |  |
| --- | --- |
| D1 | Graph |

|  |
| --- |
| 0 |
| Maze Solver |

|  |  |
| --- | --- |
| D2 | Initial Node |

|  |
| --- |
| 5 |
| Quit |

|  |  |
| --- | --- |
| D3 | Destination Node |

|  |
| --- |
| 4 |
| Find Path |

Blue arrows mean the process it comes from uses the data it points to

Black arrows mean the process creates/updates the data it points to

|  |  |
| --- | --- |
| D4 | Nodes Visited |

|  |  |
| --- | --- |
| D5 | Nodes of Path |

**Testing**

● Test that the program displays a welcome message when the program starts

− Start the program and look for the welcome message to be displayed

● Test when a non-existing file is inputted when configuring the graph

− An error message should be displayed and the user should be re-prompted for a file name

● Test when a file is inputted that the node is created when configuring the graph

− Use a small, 3 line .txt file and have the program display the nodes and their connected neighbors

● Test that the program displays that the program was “Unsuccessful: No path can be found” when an existing starting node and a non-existing destination is inputted

− Use a small, 3 line .txt file

● Test that the program correctly displays the shortest path when existing nodes are entered for starting positions and destinations

− Use a small, 3 line .txt file

− Use the provided example .txt and the example conditions on the “Lab 3.pdf”

● Test that the program displays a than you message and ends when the user quits the program

− Once the program is running, type “Quit” after “Enter the name of the Maze configuration file:”. Watch for the ending message and for the program to stop running