Directed Graph Implementation

Cpt S 223 Homework Assignment

by Evan Olds

Submission Instructions:

Submit source code (zipped) to Angel BEFORE the due date/time. If the Angel submission is not working, then submit to TA via email BEFORE the due date/time.

Optional: Include a readme.txt file in the zip with any relevant information that you want the grader to be aware of.

Assignment Instructions:

Read all the instructions *carefully* before you write any code.

Download the zip file from Angel and open the Visual Studio project included within it. Do not create a new project. Open the existing one from the zip. Complete the implementation of the DirectedGraph class functions listed below. For each function, a brief explanation is provided here. Look at the comments in the project code for more information.

```
bool DirectedGraph::AddNode(const string& nodeName,
const string& nodeData)
```

Adds a node to the graph. Do not allow two nodes with the same name in your graph.

```
bool DirectedGraph::AddEdge(const string&
sourceNodeName,
    const string& targetNodeName)
```

Adds an edge to the graph. Only add the edge if both the source and target node already exist in the graph.

```
int DirectedGraph::GetDegree(const string& nodeName)
```

Gets the degree of the specified node.

```
bool DirectedGraph::GetShortestPath(const string&
startNode, const string& endNode,
     bool nodeDataInsteadOfName, vector<string>&
traversalList)
```

Gets a list of either node names or node data values for traversal of the shortest path from the specified start to end nodes.

```
bool DirectedGraph::NodeExists(const string& nodeName)
```

Gets a Boolean value indicating whether or not a node with the specified name exists in the graph.

Again, see comments in the code for each of these functions for more details.

The graph keeps two dynamic arrays (vector objects). The first vector stores pointers to the all the nodes in the graph (m_nodes) and the other stores pointers to all the edges in the graph (m_edges).

Nodes in the graph have two strings declared in them: Name and Data. The Name string is intended to be a unique identifier for the node, which is why your AddNode function must never add two nodes with the same name to the graph.

Edges in the graph refer to the nodes they connect by *index*, not by keeping actual Node pointers. The index values in the edges correspond to the nodes in the m_nodes vector.

Nodes also have lists of incoming and outgoing edges for efficiency purposes. So while the m_edges vector is your "master list" of all edges in the graph, you also have lists of edge pointers in each node. Remember to add values appropriately to all these lists in your node addition function.

<u>Testing</u>:

Use the input file "in1.txt" included in the zip and make sure your output matches the solution in the file "out1.txt". Note that if one of your shortest paths is different, then it ***might*** still be correct but it must be:

1. Of the same length as the shortest path from the solution output
2. A valid path that can be traversed given the graph's declaration (open up in.txt and read through it if you need to verify this)