CS 260
Programming Assignment 8

This lab is worth a total of 100 points; 10 points for the self-evaluation, 60 points for the basic lab, and 30 points for the advanced lab.

This lab should follow the course coding requirements and be split into multiple files as per your chosen language. There is a driver provided for this lab. Test the code as it is developed.

Base Lab

Create a directed graph class named DGraph which can store information about a graph containing a maximum of 20 nodes. It should maintain two arrays.

The first, **G**, stores each node in the graph at a separate index. It is a 1D array of nodes, where each node consists of a value, a variable tracking whether it's been visited, and an adjacency list containing the neighbors it can reach. Because the adjacency list is a linked list, it should be comprised of edges (like nodes in previous assignments), where each edge contains the value of the neighbor and a link to the next edge in the list. You may assume all node values in G are unique.

The second array, **M**, is an adjacency matrix for the graph. More specifically, it's a 2D array of integers.

Create the methods below. You can find pseudocode and/or examples of how they work in the documents posted in Moodle for this week.

class DGraph

- constructor (init) – Set number of nodes in the graph to zero, and initialize adjacency matrix to contain all zeros.
- addNode (value) – Add a new node to G containing value, mark it as unvisited, and initialize its adjacency list to nullptr/null/none. Throw an error if the graph is full.
- addEdge (starts, ends) – Find the location in G of the node with value starts, and add a new edge to its adjacency list with the value ends. Throw an exception if either starts or ends is not in G or if they equal each other. Update the appropriate location of the adjacency matrix to indicate there is an edge from starts to ends.
- listNodes ( ) – Return a list of nodes stored in G in the order they were added.
- displayAdjacency ( ) – For each node in G, on a separate line, display the node's value, followed by a colon and a list of adjacent nodes in the graph which it can reach
- displayMatrix ( ) – Display the adjacency matrix using a '.' when there is no connection and a 1 when there is a connection.
- breadthFirst (value, edge, unreachable) – Mark all nodes as unvisited. Starting at the node containing value in G, perform a breadth first traversal, printing out the node values in the order they're visited. For the first node, print its value followed by a colon (e.g., A:). For all remaining nodes, if edge is false, simply print the node's value. Otherwise, if edge is true, display the full edge (e.g., A-B if the current node is A and the next node to visit is B). After the traversal, if unreachable is true, print out all nodes in G which did not get visited. Set edge to have a default value of false and unreachable to have a default value of true.

- depthFirst (value, edge, unreachable) – The steps are the same as for breadthFirst, except that a depth first traversal is employed.

Optionally, you may want to add the following helper functions to the DGraph class.

- findNode (value) – Returns the index in G of the node which contains value.
- resetVisited ( ) – Mark all nodes as unvisited.

Use the provided driver to test the DGraph class and verify your output matches the output expected by the driver.

Advanced Lab

For the advanced lab, add the following two methods to the DGraph class and test them with the provided driver.

Minimum Spanning Tree

Add a method to DGraph named minTree which accepts a starting node as a parameter and returns a string showing a minimum spanning tree with the starting node as root. This algorithm should use the depthFirst traversal and send the appropriate values for edge and unreachable to produce output in the format shown below. Note this is just an example and not based on data in the driver.

A: A-B B-C C-D B-E B-F F-G

The starting node is before the colon and the edges shown after the colon are in the order they were visited during traversal.

Connectivity Table

Add a method to DGraph named connectTable which returns a string containing the connectivity table for G. To solve this, remember that a breadth (or depth) first traversal will report all nodes which can be reached from a given starting node. Using this information, consider how to show connectivity for all nodes in G by starting a traversal at each node in turn.

The output connectivity table should be formatted as shown below.

```
A: C D E
B: D
C:
D: A E
E: B
```

The first letter indicates the starting node and the letters after the colon show the nodes which can be reached from that starting node. Note this is just an example. The driver will show what the proper output is for the graph implemented there. Be sure to use a breadth first traversal, if you want to match the expected output in the driver more closely.