**COMP3270 Section 002 Fall 2014, Programming Assignment 2**

This lab requires you to implement BFS and DFS, using a set of provided classes for the graph, for nodes (vertices), and for edges.

I have provided a main program for you that reads the input from standard in, and writes output to standard out.  The main program is defined in file**prog2.java**in the files section of  Canvas.  This class includes code to read the input and construct an instance of the **Graph**class containing the nodes and vertices specified by the input file. The format of the input file is  a set of lines containing two integers. The first line’s first integer, say n, is the number of nodes in the graph (to be numbered 1 to n), and the second integer, say m, is the number of edges.  Following the first line are m more lines, one for each edge. The first number on an edge line is the number of the vertex (node) at the tail of the edge, and the second number is the number of the node at the head of the edge.  Several test files with names like **testfile1.input, testfile2.input** are provided for your testing.

You are to implement methods **BFS, DFS**, and **DFS\_VISIT** in the file prog2assgign.java. Each one should implement the pseudo-code algorithm of the same name from the text book.  Do not modify any of the other methods in class **prog2,**class **Graph**, class **Node,** or class **Edge**(the last 3 are found in files **Graph.java, Node.java, and Edge.java** in the files section of Canvas).

You will observe that the node list for the graph, and the adjacency list for each node is defined using a Java **ArrayList**. To make up for the fact that the book numbers vertices starting with 1, and the Java ArrayList numbers its elements starting with 0, a dummy node numbered zero is added to the **nodeList** data member for the class **Graph.**This node is not counted in the **numNodes**. This practice allows the nice property that you can get the number of a node by using the ArrayList method **indexOf.**

In several places in the algorithm you must iterate over all the nodes in the graph, or all the nodes in the adjacency list of another node. The java construct

**for (Node n: G.nodeList) {**

**}**

allows you to efficiently do this. Note that you must NOT process node 0 if you do that.

 In BFS you must implement a queue data structure to keep track of nodes discovered but not yet processed. While Java has a built-in class**java.util.Queue,** it is merely an interface. However, the class **java.util.LinkedList**will serve for that  purpose (already **import**’ed for your convenience).

 Submit only your prog2.java file, renamed prog2\_username.java where "username" is replaced by your AU username, e.g. "prog2\_roc0004.java" (note you will have to of course rename the class defined in that file also).