CSC 323 Algorithm Design and Analysis, Spring 2018
Instructor: Dr. Natarajan Meghanathan

Project 7: Use the Results of Depth First Search to Assign Directions to an Undirected Graph and Obtain a Directed Acyclic Graph (DAG)

Due: April 12, 2018: by 1 PM (in Canvas)

The objective of this project is to use the Depth First Search (DFS) algorithm to assign directions to the edges of an undirected graph so that the resulting directed graph is a directed acyclic graph (DAG). You are given the code to run the DFS on an undirected graph as well as all the auxiliary classes needed. The Graph class has member variables to assign the push order and pop order for the vertices as well as to keep track of the number of vertices that have been pushed and popped until then.

Strategy: Run DFS on the given undirected graph and determine the pop order of the vertices. For any undirected edge u-v in the graph, assign the direction u -> v if the pop order of u is greater than the pop order of v, or assign the direction v -> u if the pop order of v is greater than the pop order of u. You are given the code for running the DFS algorithm (in a recursive fashion) on an undirected graph (whose edge information is input to the user).

Your tasks in this project are as follows (Submit as one word or PDF document):

(1 - 60 pts) Extend the DFS algorithm to determine/set the push and pop order of the vertices and extend the code in the main function to use the pop order of the vertices in the graph and assign the directions to the edges. Your code in the main function should output the push and pop order of the vertices as well as the directions assigned for the edges so that the resulting directed graph is a DAG. **Include the entire code with all the extensions in your project report.**
(2 - 15 pts) Justify why the above suggested strategy of assigning edges will indeed work. Give a clear explanation.
(3 - 15 pts) Draw an undirected graph of 10 or more vertices and use the above strategy to assign directions of the edges.
(4 - 10 pts) Prepare the edge text file based on the undirected graph of (3), pass it to your code of (1) and capture a screenshot of the output. Compare the directions of the edges assigned in (3) and (4): they are expected to be the same.

A sample output for an undirected graph of 8 vertices is shown below.

```
Enter the file name for the edges of the graph: graphEdges_2.txt
Enter number of nodes: 8

Push and Pop Order of 0 : 1 8
Push and Pop Order of 1 : 2 7
Push and Pop Order of 2 : 3 6
Push and Pop Order of 3 : 5 3
Push and Pop Order of 4 : 4 4
Push and Pop Order of 5 : 6 2
Push and Pop Order of 6 : 7 1
Push and Pop Order of 7 : 8 5

Directions assigned to the edges
0->1
0->7
1->2
1->3
2->4
2->7
3->5
4->3
4->6
5->6
```