LAB SESSION 5

GRAPH

1. OBJECTIVE
The objectives of Lab 5 are (1) to introduce an implementation of graph in C++ and (2) to practice algorithms on graphs.

2. INITIAL CODE
In this lab, students are provided initial code of a graph implementation in the files `Graph.h` and `Graph.cpp`. In this implementation, we use an adjacency matrix to hold the information of the edges. Information of a vertex is simply represented by char indicating the vertex label.

In the initial code, students will also find some basic methods to add and remove vertices already fully implemented. In addition, full implementations of stack and queue are also provided for your convenience.

3. EXPERIMENTS AND EXERCISES
Accomplish the following tasks:

3.1.
   a. Use the pre-implemented methods `addVertex` and `addEdge` to build a graph as presented in Figure 1.
   b. Call method `dfs` to display the graph vertices.
   c. Delete vertices labeled ‘6’ and ‘3’ in the graph and call the method `dfs` again to observe the result.

3.2
   a. Modify method `dfs` to display indegrees off all vertices in depth-first traversal order.
   b. Accomplish the incomplete method `outdegree`.

Figure 1

CuuDuongThanCong.com  https://fb.com/tailieudientucntt
c. Accomplish the incomplete method \textit{bfs} (Hint: you may want to use the provided Queue class)

d. Resolve the exercise 3.2a but \textit{bfs} is used to display graphs instead of \textit{dfs}.

3.3 Write additional method \textit{displayTopo} to display all of vertices of the graph in a topological order.

3.4. Implement the \textbf{simulate} algorithm in tutorial 5, question 5 as an additional method of class \textit{Graph}.

3.5. Write an additional method to find the path from the source to destination such that the weight of the path is minimal. (Students can choose an appropriate structure to store the found path).

3.6. Find the path from the source to destination such that the weight of the path is maximal but this weight must be smaller than an input number.