CSUC CSCI 311 - Algorithms and Data Structures
Lab Assignment 7

Goal(s): To practice using Graph.
Our edges will be undirected and unweighted and we will assume that nodes are labeled 0 to n-1 where the total number of nodes in the graph is n. The main goal is to implement the depth- and breadth-first search algorithms for these structures. Pay particular attention to how the graph is represented. The vector of nodes should be organized so that indices correspond to node IDs. However, indices of neighbors may not correspond to node IDs. In particular, for a graph G, G.nodes[i]->id is i but G.nodes[i]->neighbors[j]->id is not necessarily j. This is essentially an adjacency list.

Submission: C++ solutions to these problems should be written in a file called Graph.cpp (a skeleton is available on Canvas along with GraphDriver.cpp) and Submitted on inginious.

Coding Style: Note that your submission should use good C++ coding style and should include appropriate comments (for readability and maintainability). Specifically, your code must follow common C++ coding conventions for Naming, Indentation, and Comments. Points will be deducted if these are not present.

Collaboration: There will be time in lab to discuss these problems in small groups and I highly encourage you to collaborate with one another outside of class. However, you must write up your own solutions **independently** of one another. In addition, do not post solutions in any way. Also, please include a list of the people you work with in a comment section at the top of your submission.

Have fun!

Assignment Date:	**Apr 15, 2024**
Due Date:	**11:59pm on Apr 21, 2024**
Grade Date: Date:	**11:59pm on Apr 24, 2024**

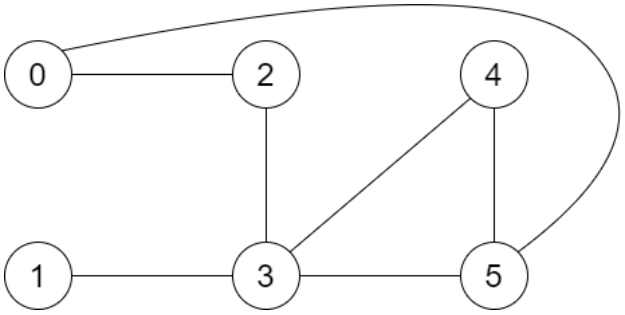
Grading Notes: 1) Assignment is due on the Due Date. 2) For each day late after the Due Date, there will be 10% penalty on the assignment's grades. 3) Submission is not accepted after the Grace Date. In other words, you will receive zero pts if your submission is not received by the Grace Date.

Grading: Coding Style 5 pts, Test Cases 95 pts, Total 100 pts.

Assignment Questions (95 pts)

1. Implement a default constructor for the Graph class. Initially, there should be no nodes in the graph.
2. Implement the printAdjList() method. This method prints information about the graph in the form of an adjacency list. Information for nodes should be printed on separate lines starting with node 0 and ending with node $n-1$. Each Line should start with a node id followed by a colon and a space separate list of adjacent nodes. The order of nodes is determined by the order in which the neighbors of each node are stored. For example, for the graph below, printAdjList() should produce the output:

0: 2 5
1: 3
2: 0 3
3: 1 2 4 5
4: 3 5
5: 0 3 4



3. Implement the isNeighbor(int u, int v) method. This method should return true if v is a neighbor of u and false otherwise.
4. Implement the DFS() method. This method should prepare the graph for a depth- first search and should call the recursive method DFSVisit(int s, int time) for nodes with increasing ids.
5. Implement the new DFSVisit(int s, int time) method. This method runs a depth-first search from node s with start time time (the discover time of s should be time + 1). It keeps track of discovery and finish times for each node. The discovery time for the first node visited should be 1.
6. Implement the BFS(int s) method. This method runs a breadth-first search from node s updating the dist attribute appropriately.
7. Implement the distancesFrom(int s) method. This method uses BFS(int s) to determine shortest path distances to all nodes in the graph from s.