CSUC CSCI 311 - Algorithms and Data Structures
Lab Assignment 8

Goal(s): To continue to practice using Graph.

In this lab we will extend our Graph class with a few small methods useful for describing features of the graph structure. Once again, 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. Some of the methods in this lab will depend on working BFS or DFS implementations.

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:	**Nov 17, 2025**
Due Date:	**11:59pm on Dec 02, 2025**
Grace Due Date:	**11:59pm on Dec 05, 2025**

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 Due Date.

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

Assignment Questions (95 pts)

1. Implement the `bool isTwoColorable()` method. This method should return `true` if the vertices of the graph can each be assigned one of two colors such that no pair of adjacent nodes have the same color and `false` otherwise. Graph coloring is a very interesting question in general with important connections to a variety of applications including scheduling and resource allocation.

2. Implement the `bool isConnected()` method. This method should return `true` if the graph is connected and `false` otherwise. Recall that an undirected graph $G = (V, E)$ is connected if, for all pairs of vertices $u, v \in V$, there is at least one path between u and v.

3. Implement the `bool hasCycle()` method. This method should return `true` if the graph has at least one cycle and `false` otherwise. Recall that an undirected graph $G = (V, E)$ has a cycle if there exist vertices $u, v \in V$ such that there are at least two distinct paths between u and v.

   The method `bool hasCycleRecur(int s)` might be helpful here. You are not required to implement this method. It is intended to serve a similar purpose as `DFSVisit`.

4. Implement the `bool isReachable(int u, int v)` method. This method should return `true` if node `v` is reachable from node `u`. Note that, since we are dealing with undirected graphs, if `v` is reachable from `u`, `u` is reachable from `v` as well. In general, `v` is reachable from `u` if there is a path from `u` to `v` in the graph.