CSci 115 Spring 2016
Lab 1
(due Tue Feb 2, 6:00pm)

The purpose of this first lab is to remind you of several topics you learned in CSci 41 and to get you familiar with the environmnet you will be using for later labs. In particular, this lab involves:

- classes, members, and methods
- separate compilation (header files, implementation files, driver files)
- one and two-dimensional arrays
- linked lists (just initialization, insertion, and traversal)

The file `graph.h` defines a class for representing graphs, along with some operations on them, including the following public methods:

- `init(n)`: initializes graph to have n vertices and 0 edges
- `num_vertices()`: returns the number of vertices in the graph
- `num_edges()`: returns the number of edges in the graph
- `insert_edge(v1,v2)`: inserts a directed edge from v1 to v2 into the graph
- `list_all_edges()`: lists (to `stdout`) all edges of the graph in some order; an edge from v1 to v2 is listed as "(v1,v2)"
- `list_all_neighbors(v)`: lists all neighbors of the vertex v (i.e., all vertices w such that there is an edge from v to w) in some order; neighbors are listed on one line separated by spaces
- `no_incoming()`: list all vertices that have no incoming edges (i.e., all vertices v such that there are no edges into v)

The file `driver.cpp` defines a simple menu-based driver program that allows a user to specify a graph by incrementally adding edges and then calling these main functions on it, reporting the results. It is defined in terms of the abstract interface and #includes `graph.h`. It can be compiled separately from any particular implementation of the graph class.

You are to implement this graph class twice, once using an adjacency matrix, once using adjacency lists. In each case, you will provide implementations of all the public methods in terms of the private data. In the case of the adjacency matrix, this is a two-dimensional boolean array indexed by the vertices, and in the case of the adjacency lists, this is a one-dimensional array of linked lists indexed by the vertices, where each linked list contains the vertices that are adjacent to the index vertex. The nodes in these (singly-) linked lists just contain a vertex (an integer), and a next pointer, which is NULL at the end of the list. New edges are always added at the beginning of the apropriate list. Your method implementations should catch and report the following three errors, but otherwise continue operating:

- initializing a graph to have a negative number of vertices
- initializing a graph that has already been initialized
- performing an operation (other than initialization) on a graph that has not yet been initialized
- inserting an edge from and/or to a vertex that doesn't exist
- trying to list the neighbors of a vertex that doesn't exist.

Implement your two versions of the graph class in files `adj_matrix.cpp` and `adj_list.cpp`.

These should each #include `graph.h` and be separately compilable. We should be able to test your implementations by compiling them individually and then linking them with our already-compiled driver.

To turn in your lab, simply place your two files `adj_matrix.cpp` and `adj_list.cpp` in the `turn-in` directory in your home directory on mulan. At exactly the lab due-date (next Tuesday at 6:00pm), a snap shot will be taken of each of your turn-in directories. To test your implementations, we will compile your two files and then link them one at a time to our already-compiled driver program and run a series of tests on them. You must make sure that your program will work when compiled and linked in this way.

If you have any questions on this assignment outside of the lab period, please ask them on Piazza, so that any clarifications we make will be available to all students simultaneously.