Programming Assignment #5
CS 163 Data Structures

Submit your assignment to the D2L Dropbox (sign on via d2l.pdx.edu)

Goal of Program 5: The goal of this program is to explore graphs using an adjacency list (an array of vertices where each element has a head pointer to a LLL of edges for adjacent vertices).

INSTEAD of writing an efficiency writeup for program #5, write about the following instead:

1) In your own words, define the following terms as they relate to graphs; this will help prepare you for the final exam:
   i. Cycle
   ii. Path
   iii. Weighted Graph
   iv. Edge List
   v. Connected Graph
   vi. Complete Graph

2) Write the algorithm for the depth first traversal, using recursion. You may write it in English, C++, or pseudo-code

Programming:

Application: You have decided to help the contact tracers to map out everyone people have been in contact with. Each person would be a "vertex" and each chain of contact would be their "edge list". For example, yesterday my family came to visit unexpectantly. Now I have been in contact with six people after 19 weeks of isolation. These contacts (name and phone number) would be in my edge list. At least four of them, in turn, were at a party the day before with 7 people. So, each of their edge lists would have me and 7 others (as well as each other).

Your primary job in this assignment is to experience creating your own adjacency list based on the set of contacts. The adjacency list will be an array of vertex objects; each with a name and phone number along with a head pointer for each linear linked list representing the edge list. There is no data with the edges (it is not a weighted graph).

We will be implementing only the following algorithms (e.g., member functions);

a) **Construct an adjacency list (**An array of vertices, where each has a head pointer to a LLL of nodes. The nodes indicate which vertices are adjacent.)
b) **Insert** a connection between two vertices (i.e., inserting a node into the edge list). This would be to specify who a particular person was in contact with.
c) **Traverse** the adjacency list, displaying all of the people a particular individual has been in contact with.
d) Destroy, deallocating all dynamic memory (e.g., destructor)
*** THERE IS NO REQUIREMENT FOR INDIVIDUAL DELETE FUNCTION!