Programming Assignment #5
CS 163 Data Structures

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

Programming – Goal and Data Structures: The goal of this program is to create a graph abstraction using an adjacency list (an array of vertices where each element has a vertex and a head pointer to a LLL of edges for adjacent vertices). *Since it is near the end of the term please keep this simple!*

Application: You have decided to map out the courses you need to take in order to graduate with all of the pre-req structure for each class so you aren't shocked at the end that you have to take a class with a pre-req that wasn't yet taken! We will do this with a **directed graph**.

Your primary job in this assignment is to experience creating your own adjacency list based on the set of courses that need to be taken, as specified by the client program. The adjacency list will be an array of vertex objects; eac and a head pointer for each linear linked list representing the edge list. Each vertex is a course that needs to be taken. 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 the pre-req structure for two classes (so to CS162 is the pre-req for CS163)
c) **Traverse** the adjacency list, displaying all of the courses that can be taken as soon as I've taken this pre-req. (So, once CS162 is taken, you can take CS163 and CS201). *Do this recursively!*
d) **EXTRA CREDIT** for an implementation of a **breadth first algorithm**
e) Destroy, deallocating all dynamic memory (e.g., destructor)
f) *** THERE IS NO REQUIREMENT FOR INDIVIDUAL DELETE FUNCTION!

Things you should know...as part of your program:

1) Do not use statically allocated arrays in your classes or structures. All memory must be dynamically allocated and kept to a minimum!
2) All data members in a class must be private
3) As usual, one function should be done recursively!
4) None of your public member functions should have "node" data types as arguments. However, you SHOULD have private RECURSIVE member functions that do take node pointers as arguments
5) Global variables are not allowed in CS163 – not even in your main
5) Do not use the String class – not even in your test suite! (use arrays of characters instead!)
6) Use modular design, separating the .h files from the .cpp files.
7) Use the iostream library for all I/O; do not use stdio.h.
8) Make sure to define a constructor and destructor for your class. Your destructor must deallocate all dynamically allocated memory.
9) Remember that 20% of each program's grade is based on a written discussion of the design. *Take a look at the style sheet which gives instruction on the topics that your write-up needs to cover.*