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 head pointer to a LLL of edges for adjacent vertices). We will be implementing only the following algorithms:

a) **Build 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 node into the edge list
c) **Traverse** the adjacency list, displaying which vertices are adjacent. And, display all.
d) **EXTRA CREDIT** for an implementation of a **depth first algorithm**
e) Destroy, deallocating all dynamic memory
f) *** THERE IS NO REQUIREMENT FOR INDIVIDUAL DELETE FUNCTION!

Application: Think about what happens when you play a game. At various points, regardless of the game, the player is faced with decisions. Sometimes there are just two decisions but in more realistic games a player could be faced with a variety. For example, do I open the chest, go through the door, or turn around? Each will open up the next set of opportunities or end the game. When creating a game, a game designer needs to have planned each of these paths and they may be more complex than a tree can represent. So, a graph is a great data structure to help with game design.

Your job for this assignment is to be a "game designer" where you are going to begin planning the basics of a brand new game. You want to think about what the player will be doing as they go through the game (it could be a video game or a board game) and place (vertex) the player will be faced with multiple choices that they will get to choose. The information in the edges will represent the choices and they will take the player to the next "place" (vertex). For example, I'm at a place in the game and edge #1 allows me to open the door and it takes me to the end of the game at the next vertex.

The adjacency list will be an array of vertex objects and a head pointer for each linear linked list representing the edge list. **Create the code to allocate an "adjacency list" for a graph. The adjacency list should contain:**

(1) Vertex Information – A place in the game

(2) Head pointer (to an Edge List)

(3) Visit indicator (optional – only needed for extra credit)

(4) Edge node – Contains the information about what the player will be considering (e.g. Open the door) and then a pointer to the vertex that is connected to this choice.

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) Never perform input operations from your ADT in CS163
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.*