Programming Assignment #5
CS 163 Data Structures

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

Task 1. **Create 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.)
Task 2. **Insert** a new vertex into the graph (Only can do this if there is still room available in the array.)
Task 3. **Insert** a connection between two vertices (i.e., inserting a node into the edge list); this assumes the vertices have already been inserted)
Task 4. Destroy, deallocating all dynamic memory (e.g., destructor)
Task 5. **DISPLAY** the adjacency list, traversing through the edge list for each vertex.
Task 6. **EXTRA CREDIT** for an implementation of a **breadth first algorithm implemented with recursion**

Background -
We all have our favorite short cuts. I have learned with the construction going on with Highway 217, that knowing alternate routes is really important. I have a map of the ones I most like to use which are faster rather than slower!

This got me thinking about graphs. Wouldn't it be great if we mapped our favorite routes that we like to take – either by car, by bike or walking.

Your job for this assignment is to build a graph that allows connections to be made for your favorite paths. Each intersection where a decision is made as to which direction to take is a vertex. The edge list would represent all of the different directions you can go from this point. This would be an weighted graph (with the names of the streets or trails is the data on the edge). It can be a directed graph if you are going to support one-way streets.

The primary goal is to experience creating your own adjacency list. The adjacency list will be an array of vertex objects and a head pointers 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 (an intersection)

(2) Head pointer (to an Edge List)

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

(4) Edge node – Contains the index or pointer to the related Vertex and the name of the street or trail

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

1) Important: In this assignment you are encouraged to explore the use of the STL String class instead of using arrays of characters

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

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.*

CS163 - Checklist

Cycle	By Tuesday	Wednesday	Thursday	Friday
Single Week Cycle:	Discussion (Planning Design) and Draft .h file		Discussion Response	Finished Assignment & Writeup

_____1 **By Tuesday - Begin by Collaborating with your Virtual Group**

- Using the Canvas Discussions
- You may share your design ideas:
  - Explore sample prototypes that meet the rules of Data Abstraction and share those among your group
  - **Make sure to ask a question to help others respond to you!**

_____2 **Tuesday - Submit** a **Draft .h file and a Draft .cpp file**

- Due **by 7pm**
- Submit to **Assignments** on Canvas
- Implement and demonstrate Tasks #1-3
- Provide "stubs" for the rest of the functions
- Have header comments (1 paragraph minimum) for each file

_____3 **By Thursday – Constructive Response to your Virtual Group**

- This concludes your virtual group work!

_____4 **Friday- Submit** a **Completed Assignment with Efficiency Writeup**

- Due **by 7pm**
- Submit to **Assignments** on Canvas
- Tar or Zip all of the .h and .cpp files together
- Upload the Efficiency Writeup separately