Assignment #5
CS 163 Data Structures

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

****Assignments in CS163 consist of written homework and programming****

1) **Use Unix Tools.** For assignment #5 select two different unix tools from the following list and write a paragraph about how you were able to use them and what they accomplished for you. Pick two from this list: **diff, grep, awk, ddd, cat, man, more.**

2) **Ethics.** Think about the error messages that you have received when compiling your assignments #4 or #5. List three error messages which were hard to decipher and indicate what you think they meant. Then for each create an error message that would have been more user friendly.

3) **Evaluate.** Discuss the strengths and weaknesses of these data structures, abstractions, and algorithms:

   a) A heap implemented using a complete binary tree
   b) The heapsort
   c) Depth first versus Breadth first
   d) The order of items in an adjacency list's edge list

4) **Programming – Goal and Data Structures:** The goal of this program is to create a weighted 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).

   You have been hired to plan out a race much like the Hood to Coast race in August. The idea of the event is to take race walkers and runners through a course from start to end. But, which streets to use? There are many possible paths. So, they hired you to create a graph of the possible routes and outline the various possibilities.

   Your assignment will be to take as input a set of streets from the start to the finish of the race. The vertices should be where streets intersect. The edges will contain the data (the street name, the number of miles on this street from one intersection to the next, and the difficulty (is it a steep hill?). Since it is a race, this weighted graph is also directional (You can go from A to B but not vice versa).

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

(2) Head pointer (to an Edge List)

(3) Visit indicator (optional)

You must support (a) insert, (b) retrieve all vertices that are adjacent to a given vertex – for all of the different directions you can go at a particular intersection, and (c) display all using depth first traversal (use RECURSION). Your goal is to final all possible routes from start to finish. Each of these must be implemented recursively.

Remember, retrieve still needs to supply back to the calling routine information about the item(s) that match. Retrieve, since it is an ADT operation, should not correspond with the user (i.e., it should not prompt, echo, input, or output data).

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.*