Programming Assignment #5
CS 163 Data Structures

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

Programming Assignment #5 may not be turned in late

Goal and Data Structures: The goal of this program is to create a directed 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).

Have you ever gone to Disneyland or a similar amusement park? When they are crowded you could find yourself waiting all day in just the Dumbo line (I have spent over an hour in that line just to find out my daughter didn't want to ride it after all!). We've found the best way to do busy amusement parks is to plan the route to take based on the time of day and whether or not a parade is taking place (when there is a parade, the lines lengths shrink! Or, if it is dinner time, best be on a ride and eat later.)

Your assignment will be to take as input the different rides at an amusement part (e.g., Disneyland or Oaks park). Each ride is a vertex. The path from one ride to another consists of a set of edges through rides, eating establishments, or souvenir store.

With this information, your program needs to build an adjacency list. 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. Load the vertices and edge information into the adjacency list. This information can come from the user or an external file, your choice.**

The adjacency list should contain:

(1) Vertex Information
(2) Head pointer (to an Edge List)
(3) Visit indicator (optional)

You must support (a) build the graph, (b) display all vertices that are adjacent to a given vertex – providing all possible rides to go to when done with the current ride, directly (c) display the route from the current ride to the next ride the user wants to visit, using depth first traversal, and (d) deallocate all. The display functions must be implemented recursively.

*** 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) 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
4) 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.*