Programming Assignment #5
CS 163 Data Structures

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

NO LATE Programming Assignment #5's

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).

What are you doing for the summer? Do you have plans? Maybe a summer job? My daughter is a manager of a college painting crew. With painting they have a specific order of tasks that need to get done. Some things can be done in parallel, but most require that one task be completed before starting the next. For example, first the house must be scraped before it can be sanded. Of course some houses don't need these two steps if the paint is currently in fair condition. It may only need to be washed before moving on to the priming step. In parallel with this can come masking the windows and gutters to avoid overspray. Then comes priming and ultimately painting. This can be applied to any project (even my greenhouse project!)

Your assignment will be to take as input the different tasks that need to get done. Each task is a vertex. If the task depends on something else having been done first, then there will be a directed edge from the first task to the second.

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 tasks that can happen once this task is completed (c) display all of the tasks, 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.*