# Programming Assignment #5
# CS 260 Data Structures

**No one week grace period for this assignment.**

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

**TRY TO KEEP THIS **SIMPLE** – the goal is to get a feel for how graphs can be created and traversed but we have limited time **

You have decided to create software for event planning. It could be to plan the Thanksgiving feast or a party. It could be to plan a wedding or celebration of some sort. For a celebration with guests you would need to (1) figure our how many guests there will be approximately, (2) find and reserve a venue, and (3) decide upon a date. Once this is done you can start the design, printing, and mailing of the invitations. In parallel with this you can start the design and printing of the program. Of course before you can print a program you have to decide what band/orchestra you will hire so that information can properly be printed in the program. Before the event takes place there are tables and chairs to set up and food to prepare and present. This is just as an example.

Your assignment will be to take as input tasks that are part of an event planning activity and then build relationships between these tasks. What makes a graph necessary in this case is that there are more complex relationships than just something that is sorted or hierarchical. We need to make sure all of the necessary tasks get done before moving on to the next activity that is dependent.

With this information, your program needs to build an adjacency list. The adjacency list will be an array of vertex objects, and **head pointers** 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 comes from an external file.

**The adjacency list should contain:**

(1) Vertex Information (the task – such as printing invitations)

(2) Head pointer (to an Edge List); the edge list just indicates all of the tasks that can take place after this current task is complete (e.g., after the invitations are printed).

After building the adjacency list, allow the client program to perform the following actions:

1. Display all other tasks that need to get done once this task has been complete. Use recursion to do this.
2. Depth First Traversal – show from start to end every path that needs to take place to make this event a success! Use a combination of iteration and recursion to implement this algorithm!

Also implement the destructor to deallocate all dynamic memory. ***There is NO requirement for an 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 CS260 – 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.