**CSCD240 Lab 11**

**This lab is classified into the homework category when computing final weighted grade.**

I have provided a zip file that contains a group of .c and .h files implementing a generic linked list. A generic linked list can be used to store any type of data after you implement once. E.g. it works with strings, double or user-defined types. **Note** that a generic linked list usually stores the same type of data in it, rather than handling both double and char in one list at the same time.

Based upon our discussion in the class regarding the source code being provided, you have to implement the function at the bottom of the file List.c, named as int removeNode(List \*alist, void \*obj); The specification for this function has been provided as inline comments on top of the function header. Please read carefully the comments.

This lab tries to get you to work with linked lists and structures that contain void pointers and function pointers.

**Requirements:**

* Before you write your code, please understand the entire piece of code that has been provided.
* You have to implement the function at the bottom of the file List.c, named as int removeNode(List \*alist, void \*obj).
* Please do not change any other part of the program, except for implementing the removeNode() function.
* No memory leak please. Include a valgrind run to ensure you are not leaking memory named cscd240Lab11Val.txt.
* No segmentation fault or core dump please!
* Write a makefile that only compiles the lastest modified files, instead of recompile everything.

**To Turn In:**

Submit a zip file

* Containing all C files and H file(s).
* Your makefile – target is lab11
* Include an output captures from running your program named cscd240Lab11output.txt.
* Include a valgrind run to ensure you are not leaking memory named cscd240Lab11Val.txt

Your zip will be named your last name first letter of your first name lab11.zip

**One of the Correct Outputs is provided on the next page. Note that your grader will probably pass in a different variable to test your code.**

Output collected from the standard out!

**------Original List:------**

**ID: 0, Name: hello-->**

**ID: 6, Name: good-->**

**ID: 5, Name: find-->**

**ID: 2, Name: ele-->**

**ID: 3, Name: dog-->**

**ID: 4, Name: cat-->**

**ID: 1, Name: band-->**

**ID: 7, Name: able**

**------After sorted:------**

**ID: 0, Name: hello-->**

**ID: 1, Name: band-->**

**ID: 2, Name: ele-->**

**ID: 3, Name: dog-->**

**ID: 4, Name: cat-->**

**ID: 5, Name: find-->**

**ID: 6, Name: good-->**

**ID: 7, Name: able**

**------After removed student with id = 2:------**

**ID: 0, Name: hello-->**

**ID: 1, Name: band-->**

**ID: 3, Name: dog-->**

**ID: 4, Name: cat-->**

**ID: 5, Name: find-->**

**ID: 6, Name: good-->**

**ID: 7, Name: able**

**------Now start to free all memory!**