Student Name: ______________________________ *Date:* ________________________

CS221: Data Structures — Programming Assignment #3

Due: At beginning of class on Thursday October 12 (25 points)

Problem Statement

You are to refine the program that reads in Students from a data file. Each student should be stored in a Student object.

This program adds on to the requirements for Program #2. Please see that assignment. This will read in the same type of data from an input file and write the results to an output file.

New Requirements

1. You are to re-implement the **StudentList** data structure/data type that stores students using a **doubly linked-list**. Store the students in the list in alphabetical order (By lastname, use firstname to further distinguish in the event there is more than 1 student with the same last name). That is, each time you add a new student to the list, you must insert it in the correct location.

2. You must implement the same operations on this new version of StudentList that were required for assignment #2. In addition, you will need to add at least two new operations to support the following requirement:

We are going to use our list as a sort of directory, or white pages of sorts. Prompt the user to look up a student by first and last name. If the matching student is found in the list, print out only that student's record. If the student is not found, you should print the two nearest students (the 1st one before and the 1st one after) to the desired student. *Note - in order for this to work, your operations are going to have to have some concept of where the search stops in the list (in the middle, at the end, etc). You can implement the printer operation as a method on the list so it has this information at its disposal.*

Some Examples - Suppose I have the following people in my list

Hank Aaron
Andruw Jones
Chipper Jones
John Smith
Josh Thomas

- If I search for *Hank Aaron*, only Hank Aaron should be displayed as found in the list
- If I search for *Mark Jones*, then Chipper Jones and John Smith should be displayed as two nearest to the desired information (before and after).
- If I search for *Jack Williams*, then John Smith and Josh Thomas should be displayed.

3. You must provide the user with the ability to delete a student from the list by providing their firstname and lastname. Only delete the student node with an exact match.

Your main test program, after it reads from the input file, should prompt the user to perform lookups and deletes until the user is ready to quit the program (i.e. use a while loop of some sort). When the user is finished, write the remaining contents of the list to the output report file as in program #2.

Other Requirements:

- Please make sure that you create a Win32 Console Application in Visual Studio 2012 or 2015.
- Make sure you create an empty project, with no pre-compiled header.
- Make sure the first comment in your program indicates which version of Visual Studio your project works with.
- Split your program code files into the appropriate .cpp and .h files for each of your classes. (Exception - your NODE class can be defined in the same file as the StudentList).
- You must write your own operations. You may not use any built-in or library sorting routines.
- IMPORTANT: I will no longer accept programs that have the following non-standard C++ feature in them:
    - You may not initialize members in the class variable declaration section. They must be initialized in the constructor. These will NOT compile in standard versions of C++
    - Ex:

```
class Student {
     double average = 0.0;    ← this assignment should not be here
     . . .
};
```

String Comparisons in C++

http://www.cplusplus.com/reference/string/string/compare/

TURN IN:

Please turn in a print out of each of your program files. Submit the electronic version of your project on CANVAS. Make sure your name and the name of the file is in the comments at the top of each program file.

Grading Requirements

- Your program must be well-commented. Comment all variables, functions and remember to have a section of comments at the top of your program that includes your name, date, course section and a description of what your program does.
- Use good variable names.
- Use good and consistent naming conventions for class members.
- Use proper code indentation to make sure your program is easy to read and understand.
- You will receive no more than 50% credit if your program does not compile.
- If your program compiles but does not execute correctly, you will receive no more than 70% credit.
- Please note that the weight of this assignment (25 vs 15 points) indicates that it will take a larger level of effort to complete.