

**Computer Science III: Data Structures and Algorithms** **Assignment 2** COMP 241 Spring 2017

**Due: Tuesday February 14th, 11:59:59 pm (90pts)**

Checkpoint: Wednesday February 8th , at 9:59:59 pm (10pts)

**RHODES FACULTY PROCESSION**

Single file, please! Keeping professors in line can be tricky business. You should see the chaos in the corridor of first floor Palmer right before an official procession to a formal Rhodes College event.

For this project, you will write a multi-source-file program that prompts for and reads professors’ last names and their years of service at Rhodes College from the keyboard. The program will continually prompt the user to enter this data until they type an “X” in the response to the name prompt. At this point, the program will print the following lists to screen:

1. In input order: The list of names and service years in the same order as read from the keyboard.
2. Ordered for procession: The list of names and service years in decreasing order by years of service.
3. Ordered for the Rhodes Directory: The list of names and service years in increasing order by last name.

If two or more professors have the same name or same number of years of service, the ordering among them doesn’t matter.

Don’t use any arrays or vectors for this program – The lists of professors should be stored as dynamically allocated linked lists. Be sure to clean up the mess afterward. In other words, delete all of the dynamic storage you use before the end of the program. No memory leaks allowed!

To implement this program you will need to store 3 different linked lists. Each time you enter a new professor and their corresponding years of service, you should enter that data into each of the linked lists. The way you insert the data into the linked lists matters!

Use at least this one function in addition to the main program: Write a function that prints a list of professors with their years of service to the screen. Of course, you should call this function three times, once for each of the required output lists

The next page shows an example of a possible session, with user input shown in ***bold italics***. Your output isn’t required to match this format exactly, but it should have all of the same information clearly displayed on the screen. Note that your program should not prompt for a number of years of service after the sentinel value has been entered.

Use your own sample data (input values) while testing and debugging your program. To start this program you should use the node.cpp, node.h, slist.cpp, and slist.h that we have been working on in class. More specifically, this code in BOX in a folder called Code. Our linked list class is called slist.zip. Download, unzip the files, and use them for this project. For this project, you should add member functions to the slist class. NOTE: You will have to change the type of data that each node stores.

Rhodes Professor Procession Line-up Program

Professor Name: ***Sanders***

Years of Service: ***9***

Professor Name: ***Gottlieb***

Years of Service: ***16***

Professor Name: ***Kirlin***

Years of Service: ***5***

Professor Name: ***Sheard***

Years of Service: ***8***

Professor Name: ***Welsh***

Years of Service: ***4***

Professor Name: ***Larkins***

Years of Service: ***2***

Professor Name: ***X***

Your original list:

Professor Sanders, 9 years of service

Professor Gottlieb, 16 years of service

Professor Kirlin, 5 years of service

Professor Sheard, 8 years of service

Professor Welsh, 4 years of service

Professor Larkins, 2 years of service

Procession order:

Professor Gottlieb, 16 years of service

Professor Sanders, 9 years of service

Professor Sheard, 8 years of service

Professor Kirlin, 5 years of service

Professor Welsh, 4 years of service

Professor Larkins, 2 years of service

Directory order:

Professor Gottlieb, 16 years of service

Professor Kirlin, 5 years of service

Professor Larkins, 2 years of service

Professor Sanders, 9 years of service

Professor Sheard, 8 years of service

Professor Welsh, 4 years of service

**Putting the code together**

For this program you should have five files: slist.h, slist.cpp, node.h, node.cpp and main.cpp**.**  main.cppshould contains your own sample data and test driver program that allows you to debug and test your program. How am I going to test your program? I’ll create my own main.cpp and test it with your code.

**Turning In Your Program**

You must write this program in C++. Before you turn in your program, you should create a folder on your computer using the naming convention in the syllabus and put all your related code into this folder. For this program you should have at least five files slist.h, slist.cpp, node.h, node.cpp and main.cpp**.**

**Checkpoint:**

Sometime before Wednesday, Feb 8that 10:00pm, you must go show the tutors (in the Barret 033, Sunday—Thursday 7pm—10pm) a program that adds faculty to a list and prints out the individual faculty members in the original order you entered them. Have the tutors write me an email ([sandersb@rhodes.edu](mailto:sandersb@rhodes.edu)) with the subject **CS241: Checkpoint <your name>** if you have completed this checkpoint.

**Extra credit**

You may go above and beyond the scope of this project for extra credit. For example you can overload operator< and/or operator> for the node class . Be sure that you document what you did for extra credit at the top of main.cpp

**C++ Code Style Guidelines**

Please see Assignment 1 for details on how your code should be commented. In particular you should make sure that your .cpp and .h files have a comment and each function is appropriately commented.