**­­­­Programming Project Report**

Name: Riley Nadwodny

Date: 2/17/2020

**Academic Integrity Statement:** I pledge that I have neither given nor received unauthorized help on this programming assignment.

**Problem Statement:**

The goal of this programming assignment is to create a linked list that is used to store information about users in the group. The particular task for this assignment was to design, implement, and test a UserNode class that stores information about each user and a UserList class that contains a linked list of UserNodes that represent the members currently in the group.

The inputs for this program are the user’s selection for which menu option they select and the input for adding a new user. To add a new user to the linked list, the user must select the “Add user” option and input the following fields: a first name, a last name, a username, a phone number, and the user’s current message count. The output for this program depends on which option the user selects; there is a function to print out the entire list of users and the user with highest message count out of all the users in the list.

**Design:**

The design of this program is a little bit complex. There are five overall parts to the program: a main.cpp file, a userNode header and .cpp file, and a userList header and .cpp file. The userNode files contain the function declarations and definitions for the functions and variables used in the userNode class, while the userList files contain the function declarations and definitions for the functions and variables used in the userList class. The userList class also contains a userNode pointer which points to the head of the linked list.

The main.cpp file is the body of the program, which contains a very basic user interface that allows the user to select different operations. There are five options for the user to select: Add a new user, update a user’s message count, find the user with the largest message count, print out the user list, and exit the program. While the user does not select the “Exit program” option, the program will continue to loop, prompting the user to select their option and input the data necessary for the selected option.

**Implementation:**

To implement the design for this program, a good portion of sample code was borrowed from labs four and five, in specific the StudentNode and StudentList classes. The userNode and userList classes in this program are heavily based upon the StudentNode and StudentList classes, and perform similar functions while having a few key differences. To adapt this code, all of the ‘Student’ methods had to be renamed to the corresponding ‘user’ class, in addition to the header files.

The development timeline for this program took the full two weeks of the assignment duration. The very first thing implemented in the program was skeleton code for the userNode and userList classes including public methods, private variables, and function definitions. Next, the userNode methods were filled out with code in order to set and get values for the variables in the userNode class. Then the userList methods were filled out in order to work with the data in the linked list, including incrementing the message count of a desired user by one, adding a new user to the linked list, finding and printing out the user with the largest message count, and printing out the entire list of users in the group.

**Testing:**

To test the program, a standardized set of commands was followed in order to test each method in the main program at least twice. First, four users are added to the list by selecting the “Add user” option, and filling out the required info for each user. Next, the option to “Find user with the largest message count” is selected to make sure the user with the largest message count is returned correctly. Then, another user is added to the list, and then the whole list of users is printed out by the program. Another user is added to the list with a larger message count than any of the existing users, and then the option to find the user with the largest message count is run again in order to make sure the new user with the larger message count is output instead of one of the pre-existing users. One more user is added with a lower message count, and then the entire list is printed out again. Lastly, the “Exit program” option is selected in order to exit the program.

During testing, there were no special cases tested. If the user desires to put garbage data into the user data fields, then they will get garbage data output back to them whenever they decide to print out the list. Everything in the program worked as expected, and sample input/output is included in a separate text file, as there was a large amount of data input and a large amount of data output. Adding it all in would extend this report by a good three to five pages at the very least.

**Conclusions:**

Overall, the result of this programming assignment was a success. The program performs as desired and returns the expected results; the exception here is if garbage is input into the program, then the program will output garbage back to the user, as there is no error checking, such as making sure the first name and last name fields are actually characters instead of digits. Next time, a lot of the same steps would be taken in designing and implementing the program, as the implementation for this program made testing the methods relatively easy and debugging the code relatively easy as well. From the beginning of the design stage to the end of implementation and testing, total time to completion for this program was about two weeks.