CSCI 2170 Open Lab Assignment 7
Electronic submission Due midnight, Monday, Nov 23rd

In a social network setting, each user is connected to many "friends". One may be able to see the friends of his or her friends. It may not be as easy to view their friends' friends' friends, or to answer questions like "am I connected to this other person through a number of friends of friends in between?". **Six degrees of separation** is the theory that everyone and everything is six or fewer steps away, by way of introduction, from any other person in the world, so that a chain of "a friend of a friend" statements can be made to connect any two people in a maximum of six steps.

Suppose we are given the entire social network in terms of the friend list of every user in the network. It is possible to ask questions like:

- Who are on X degrees of separation from me? (X is an integer >=1)
- What are the list of people that would connect one person to anther person in the network?
- What is the phone number of a person?
- …

To help answer these questions, it is necessary to build a database of all the friend information. The first step of the project is to build a list of users that are in the network. This assignment implements a pointer based ***sortedListClass***. This class should keep a list of items in ascending order. In this application, the items to be stored are records/structs containing a friend's name and phone number. Your ***sortedListClass*** should be very generic, where the item may be easily changed to any type in other applications. Use *typedef* to create an alias "ListItemType" for the struct type defined.

users.dat : The names and phone numbers of the users are kept one name per line in the data file, for example:

Mary Lou
(615) 345-9281
Patrick Lincoln
(615) 483-0928
Tom Lenon
(615) 339-0929
George Carter
(615) 382-9372
Abbey Pascal
…

Copy the data files into your own account with command:

cp ~cen/data/users.dat workspace directory/project directory/users.dat

Program requirements:
The ***sortedListClass*** should include the following methods:

1. Default constructor
2. Copy constructor
3. Destructor
4. Inserts an entry into the list. The list needs to remain sorted in alphabetical order after the insertion
    a. The item to be inserted is provided as the parameter to this function.
5. Deletes an entry from the list
    a. The item to be deleted is provided as the parameter to this function.
6. Finds and returns an item by the key value (the key value in this case is the user's name)
7. Print the entire list (print in the format : name phone-number, one user per row)
8. Returns the length of the list
9. Returns whether the list is empty

Develop a client program to test ***sortedListClass***. It should have at least the following parts:

1. Create a sortedListClass object
2. Read from "users.dat" data file the user names one at a time until the end of file. Insert the names into the list. The names will be kept in ascending order.
3. Print the list of user names in the list
    a. one "name phone-number" pair per row
    b. It should include an output that displays the number of users in the list
4. Find and print a user from the list
    a. Prompts the user to enter the user name. If the user exists in the list, print the message "This user is found" and displays the phone number of the user as well. If the user is not in the list, display the message "This user is not in the network".
5. Delete a user from the list. Perform 3 delete operations. Each time, prompt the user to enter the name of the user:
    b. If the user exists in the list, delete it from the list. Then display the current list.
    c. If the user is not in the list, display a message "This use does not exit. Deletion can not be performed".
    d. Four different test cases should be used to test your code:
        i. Delete a user from the middle
        ii. Delete a user from the end of the list
        iii. Delete a user from the front of the list
        iv. Delete a user name that does not appear in the list

Instructions to submit your program

To submit the project, log onto D2L, locate the Dropbox named "Open Lab 7". Upload the following 3 files:

- sortedListClass.h
- sortedListClass.cpp
- main.cpp

Suggested steps for developing the project:

Step 1:

Do not write the code for the entire ***sortedListClass*** at once. Develop the class in steps. In each step, add in a few methods, thoroughly test them out in the client program to make sure they work correctly before moving on to add the other methods in the class.

So, for the first step, your sortedListClass may only have the following methods:

1. Default constructor
2. Insert
3. Print the entire list

Write code for sortedListClass.h and sortedListClass.cpp files, and develop a client program that only include the following steps in the client program:

1. Create a sortedListClass object
2. Repeat until the end of the data file is reached
        Read a user name from the data file and insert the user information into the list
3. Print the list of users in the list

Build and run the program with the data file. The list of user information should be displayed in alphabetically ascending order. If you get the correct output, move onto the next step.

Step 2:
Now add the following methods to sortedListClass.h and sortedListClass.cpp:

1. Finds and returns a user name
2. Returns the length of the list
3. Returns whether the list is empty
4. Destructor

And add to the client program:

1. Find and print the user from the list
    a) Prompts the user to enter the user name. If the user exists in the list, print the message "This user is found" and his(her) phone number, otherwise, say "This user is not found"

Build and run the program. If your program is able to display the correct message according to user input, you are done with step 2.

Step 3:
Add this method to sortedListClass.h and sortedListClass.cpp:

1. Deletes an entry from the list

And add to the client program the following steps:

1. Delete a user from the list. Perform 3 delete operations. Each time, prompt the user to enter the name of the user. If the user exists in the list, delete it from the list. Otherwise, show a message "This user does not exist. Deletion can not be performed". Print the list of user names after each deletion operations. Four different test cases will be used to test your code:
    b. Delete a name from the middle
    c. Delete a name from the end of the list
    d. Delete a name from the front of the list
    e. Delete a name that does not appear in the list

Build and run your program. If your program displays the correct output according to user input, you are done with step 3.