CS 130A (Fall 2015)

Project - Part 1

Date Assigned: Monday, September 28, 2015
Due: Saturday, October 17, 2015 at 5:00 PM
Submission name for this part: Project Part1

This is the first part of your long-term Social Network project. This project can be done only by single student. For this part of the project, and all other subsequent ones, you should be using Makefiles. You should submit your code (as a zip archive) on Gauchospace before the due date. It should contain a README file with the instructions on how to compile and run your code.

As we make progress in the project, you will be reusing and improving this code a lot. We strongly recommend that you design it cleanly and comment it well. You will thank yourself in a few weeks.

Also, we recommend that you read all of the questions before beginning to work on any one. The questions build up toward a coherent whole, so how you solve one question may impact how you solve another one.

We strongly recommend that you create separate files for each of the classes you will write. Also, you should separate your header files from the code for each of them. Follow good practice in terms of coding style.

1. Review chapters 1.4, 1.5, 1.6 and Appendix A from the textbook.

2. Implement your own template doubly linked list class. Create a separate template class for list node that can store data of any type (given by template parameter). Members of the class should be private and you should create public methods for accessing and modifying the members. Your linked list class should at least have the following functions:

   (a) a constructor (or multiple, if you want) and destructor. Make sure you are not causing any memory leaks.

   (b) a function for adding an element to the end of the list

   (c) a function for removing an element from the list

   (d) a function that enables you to iterate through all elements, e.g., by returning the head element.

3. Each user in the social network gets a Wall where he/she can post something. Create a class `WallPost` for a wall post. At the least, your class should contain the following fields (which should be private):

   (a) a string that stores what the post says

   (b) time when the post was made

   (c) one more field of your choice (for ex. username of the author or something else you think of).

   Your class should have at least the following functions, which should be public:

   (a) a constructor (or multiple, if you want) and a destructor.

   (b) functions to read and modify the values of the private fields.

(c) a function that returns the entire wall post (string and the other field) as a string in a nicely formatted way, for printing.

Later in the project, you will want to add things like responses to wall posts, visibility of the post, and more. This is why creating a separate class for the post helps rather than just having some strings.

4. Create a class for maintaining the **Wall** associated with a user. This class should contain a private linked list of **WallPost** elements so that each node in the list stores a **WallPost**. Use your own implementation of doubly linked list from earlier part. In addition, you can also store **username** of the user to which this wall belongs. Wall class should have at least the following functions:

   (a) a constructor and a destructor.

   (b) a function for adding a new wall post to the wall

   (c) a function for removing a wall post from the wall

   (d) a functions to read and modify the **username**

   (e) a function for writing the entire wall with all the posts to a properly formatted string. You should reuse the function from **WallPost** class to get posts as a string.

   (f) a function for reading a wall from a properly formatted string. For now, you don't have to worry about misformatted strings. The implementation needs you to parse a given string and make this wall contain all the **WallPost**s in the string. You may need to clear this wall of all the previous posts.

5. Your social network consists of multiple users. Implement a class **User** for a user of your social network. Each user should have a Wall, and private variables for user's **username** and **password**, user's real name and at least one more item of your choice (such as birthday, city, or something else). Username for each user should be unique. You will ensure that later while adding new users. At the least, it should have the following functions:

   (a) a constructor and a destructor. You should ensure that you also set username associated with the **Wall** in the constructor.

   (b) functions to edit the user's private data field, where it makes sense.

   (c) a function for adding a wall post by the user on his/her own wall (for now, no one can post on anyone else's wall).

   (d) a function for deleting a wall post by the user on his/her wall.

   (e) a function that gives a string containing all of the user's information in a nice format. (Yes, that includes even the password for now you'll fix that later.)

   (f) a function for reading the user's data from a properly formatted string. If you want, that function can just be another constructor.

   Notice that for now, we are not storing any information about user's friends. So our social network isn't very social, or much of a network, yet. We'll fix that in later parts.

6. Implement a class **UserNetwork** for a database of users. It should contain a linked list of all the users in the network (using your own Linked List implementation). It should have the following functions:

   (a) a constructor and a destructor.

   (b) a function to add a user. When you do this, make sure not to create another user with the same username as an existing one.

(c) a function to delete a user based on the username.

(d) a function to write the entire formatted user list (including all data) to a file.

(e) a function to read an entire (correctly formatted) file and turn it into a user list.

7. Implement a simple text-based (cin, cout) user interface for your social networking site. When you start the program, it should first read all the information about existing users from a file. It should allow you to:

   (a) Create a new user. Display an error if username already exists. After the new user is created, you can either make the program exit, or return to this menu.

   (b) Log in as a user. For this, you will probably want to add functionality to your `UserNetwork` class so that you can find a particular user. When the user logs in (entering a name and password), this means that the user will enter the menu described below, and have access to his/her posts, add to them, etc.

   (c) Quit the program.

   When a user logs in, your program should allow the user to:

   (a) Display his/her entire wall.

   You will add more options in the next part of the project.

   The user interface you write can be simple, but it should be safe. Your program should not crash just because someone enters garbage at the prompts. That is, make sure to check for legal inputs. However, you don't have to guard for now against someone tampering with your data file.