Geoffrey G. Lord, Jr.
Final Project Writeup
CSCI 2270
12.8.19

User Password Protected Database

Overview and Data Structures Used:

The primary focus of this project was to create a secure system that allows for the organization of users information. Systems like this are very common and applicable to the real world because it allows all users to view information whilst insuring only the owner of an account can make changes. This project is an overly simplified version of how users information can be stored in social media applications. For example, on Facebook it is imperative that only the owner of an account has the ability to change their information. A simple way to insure that only the owner of some information has the ability to change their information is to implement a system that allows that data to be password protected. Currently, it is deemed unacceptable to store password information in plain text and therefore I have implemented a hash function that is used to convert a string of characters into a significantly large complicated number. This is done through a simple mathematical computation that uses modulus to make the reversing of the "hashed password" almost impossible. This process allows only the user to have access to their account. This system is fairly secure but could easily be cracked via a brute force attack. Another precaution that was implemented into this user database is a log system. This log contains the 10 most recent changes that have been made to the user database. While this log does not prevent a malicious actor from attempting to change the system it does allow a user to look back and see that the system may have been altered with. This system organizes users accounts by use of a Binary Search Tree. This data structure was chosen due to its performance. With my implementation the complexity can be expected to range between O(n) and O(log(n)) due to the fact that the tree is not self balancing. Additionally, I made used of a queue using a vector implementation to store the logged data. A queue was used because it allows for a first in first out structure. This unique aspect of a queue allows the log to always keep the most recent changes made. For examples sake this log only holds the last 10 changes made to the system but in a real world setting it would most likely hold many more logged actions.

Functionality:

Upon running the program the user will be faced with 6 options listed in a homepage.

```
HOMEPAGE
========
1. Add User
2. Print Users
3. Edit User
4. Delete User Info
5. View Log
6. QUIT

Please select an action (1,2,3,4,5,6):
```

Figure 1

1. Add User: This selection will allow the user to add one or many accounts that will be comprised of a username, password, location, and age.
2. Print Users: This selection allows the user to print the entire dataset to see all of the users in the tree. There is no password needed to access the list of users. An example output of this selection can be seen in figure 2.

3. Edit User: This selection will allow the user to change their location and age within the system. The user however will not be able to change their password or username. A user must provide a correct username and password to access this feature.
4. Delete User Info: This selection will allow the user to delete the data associated with their node. To access this feature a user must provide valid login credentials. Once the user has provided their credentials the system will automatically overwrite their data. This action will not delete the users node but will remove their personally identifiable information from the tree.
5. View Log: When any action is made in the program it is logged in a queue. This session log can be accessed via selection 5. A log example can be seen in figure 3.

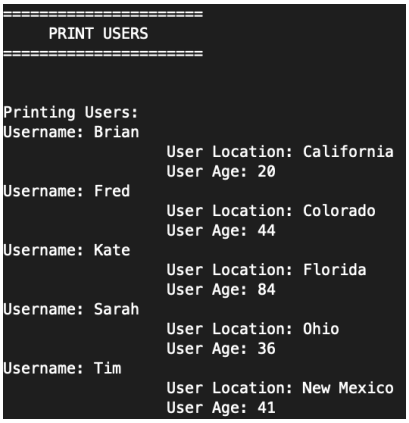

Figure 2

Figure 3

Data Used:

The data used in this system is quite simple. The BST is populated with 5 users when the program is first run. The user information is taken in through the command line and is placed into four string variables. Three of those variables (userName, location, and age) are left unchanged. The password variable is converted from a string to a double type variable and is hashed so that the password does not need to be stored in plain text.

Results:

The result of this project was a very beneficial learning exercise. It allowed me to see all of the nuances that must go into sorting and organizing data in a user database. This project also allowed me to focus on the security that is required in creating a system that should not give everyone access to everything. Furthermore, this process has allowed me to implement two of the data structures taught in the class in a very applicable way. In the future I hope to continue to work on this project and implement a functional user interface that would allow the user experience to be more streamlined and intuitive.