**CSCE 2014 – Programming Project Report**

Student Name: Natalie Pruitt

Submission Date: 11/20/2024

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

**Problem Statement:**

This project is intended to use a hash table with separate chaining to store and retrieve names and phone numbers to simulate a phone book. The user inputs a text file, and strings of names and numbers. The program has many outputs including: the final phone book, the insertion or deletion of a name, and the location of a name. Error handling is required for invalid user inputs.

**Design:**

For the user interface, I first asked the user to input a file. This file was read in using a HashTable method that opened the file, got the first and last names and phone numbers as strings using a while loop and if statement, then concatenated the first and last names as one string. After completing this, the names and numbers were inserted into the hash table. I then designed a menu for the user to choose what they wanted to do with their hash table. They could (1) insert a name and number, (2) search the table, (3) delete an entry, or (4) print the table. I had the user enter the corresponding number and had an if-else if statement allow them to do what they wanted. This if-else if statement was nested in a while loop that ran as long as the user wanted to keep modifying their table.

For the linked list class, I had insert and search methods that took two string parameters, a delete method that took a string parameter, and print, content-checking, and count methods that took no parameters. The only private variable for this class was the head. Insert(), Search(), Delete(), and IsEmtpy() were all Booleans that returned a truth value. Count() returned an integer for the number of nodes in the list. For the hash class, there were also insert, search, delete, and print methods that took the same parameters and returned similar values. This class also had a read method that took a string parameter and three private variables: a Hash() method, the size of the table, and a linked list variable.

As mentioned earlier, most of the main program (after the testing portion) occurred within a while loop as a series of if statements. If the user chose (1) on the menu, they were asked to provide a name and number, then were alerted if the entry was successfully inserted. If they chose (2), they were asked to give the first and last name. Search() was called and the user was alerted if they were found. Similarly, for (3) and Delete(), the user provided a first and last name were let know if their request was successful. Finally, for (4), Print() was called and the entire table was displayed. I decided to let the user make each request as many times as they wanted, hench the while loop. Once they were finished, they responded ‘n’ instead of ‘Y’ and the program exited.

**Implementation:**

I began with Dr. Gauch’s linked list example. I changed the parameter values to both be strings. In the main program, I read in the first 100 names of the phone.txt file and put them into the linked list, rearranging the order to ‘First Last’ and combing both names into one string. I printed this out to ensure that it worked, then I moved on to the HashTable class. Again using Dr. Gauch’s source code, I implemented the class, again changing the method parameters to strings. I added a read method and implemented it using the code I used to read in my file from my linked list test.

Back in the main program, I began to test my methods. I created a HashTable object of size 100 and read in the phone book file, printing it out to ensure correctness. I then inserted two hard-coded entries, searched for two (one I knew was there and one I knew was not), and finally deleted an entry. After all of these tests, I began working on the meat of the main program. I created all the cout’s and cin’s to get information from the user, then created my while loop and if-else if statements, calling the corresponding HashTable methods in their respective statements. I complied and ran the program, using a variety of inputs to test and debug the program until it was working as intended.

**Testing:**

I first tested all the HashTable methods using hard-coded inputs.

A screenshot of a computer

Description automatically generated

This worked as expected. Then, I tested my menu options.

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

The results were as expected. Finally, I tested an invalid menu input.

A screenshot of a computer program

Description automatically generated

The program behaved as I expected it to behave.

**Conclusions:**

This assignment successfully used linked lists and hash tables to simulate an online phone book. It allowed the user to use a menu to alter and view their hash table. In a future program, I would keep most of the program the same, especially when it comes to allowing the user to choose from the menu as many times as they would like. However, I might keep the names in ‘Last, First’ order as well as alphabetical to more accurately represent a phone book. Overall, this project took approximately 4 hours over 2 days to complete.