

CS2420 – Fall 2008
Program 3
Due September 29, 2008

HashTables

In this assignment, you will implement a program to simulate a simple password saver. The data structure for the password saver is a HashTable object. You are required to write the main program that completes tests all of the options in the HashTable class.

Requirements for the HashTable Class

- The username and password pairs must be store in a HashTable object.
- There must be methods to allow the user to:
 1. Load the password file into the HashTable object. The username and password are read as a pair from a file and the pair entry is added into the HashTable object.
 2. Add a new username and password. The username and password are read as a pair from the user and the pair is added into the HashTable object.
 3. Remove an existing user. The username is obtained from the user and the corresponding pair entry is removed from the HashTable object.
 4. Change an existing user's password. The username and password are read as a pair and the new password is put in the pair object's second entry in the HashTable object.
 5. Check if a user exists. The username and password are read as a pair from the user and checks if the pair entry exists in the HashTable object.
 6. See the structure and contents of the HashTable to screen.
 7. Obtain the size of the HashTable.
 8. Save the username and password combinations to a file.
 9. Exit the program.
- The client must reprompt the user for the next choice from the menu and exit the program only when the exit option is selected.

In addition to the above, your implementation **must at least** provide the following functionality:

- HashTable() – no-argument constructor. Initializes the vector to a default capacity of 11. We use a prime number to enhance the quality of the hashing performed later. We shall refer to the resulting capacity as 11 buckets.
- HashTable(int n): 1-parameter constructor. Initializes the vector to a specified capacity. This constructor is called by the client to set the capacity of the underlying vector to the highest prime number below n.
- ~HashTable() – destructor. Iterates through the vector and clears all the lists within each bucket.

Method that loads entries from a file: This method should read the username and password entry from a file and add the entries into the HashTable. Each pair entry is separated by a carriage return. A line entry is represented in the format: username: password

For example, a sample entry file is as follows. You can get the file from the course web page as passwd.txt:

```
hemmi:l5ltkY7/QQ.yHP1hhZrcy/  
marge:qnR/8Wr64NvX3qQUy.Q7U1  
hank:dJP.JOXf9U3VUdm8QunMy/  
toh:Czz4fmL29pz0vLPvaCr1.0  
sean:x.PthYV62/lu9ecwrGKBV0  
stuart:sN0AKSTpbrRfpePPk4Qpl/  
matthew:CjA3fXUld9/nilBzoK1pr0  
kelly:RixHNHGrE6oUZda1O/lvZ.
```

Method that removes a user: This method should find the username in the HashTable and remove the pair entry from the HashTable.

Method that changes a username's password: This method should find the pair object containing the username and password and modify the pair's second entry in the HashTable with the new password.

Method that find's a username: This method should inform the user if the username and password pair entry exists in the HashTable.

Method that prints HashTable's contents: This method should print the structure and contents of the HashTable to the screen.

Method that returns the size: This method should return the size of the HashTable.

Method that writes entries to a file: This method should write the username and password entries in the HashTable to a file. Each entry is separated by a carriage return. A line entry is represented in the format: username:encrypted_password

For example, a sample entry output file is as follows:

```
hemmi:l5ltkY7/QQ.yHP1hhZrcy/  
marge:qnR/8Wr64NvX3qQUy.Q7U1  
hank:dJP.JOXf9U3VUdm8QunMy/  
toh:Czz4fmL29pz0vLPvaCr1.0  
sean:x.PthYV62/lu9ecwrGKBV0  
stuart:sN0AKSTpbrRfpePPk4Qpl/  
matthew:CjA3fXUld9/nilBzoK1pr0  
kelly:RixHNHGrE6oUZda1O/lvZ.
```