

Github link - <https://github.com/LasithaJananaya/CS2023-Data-Structures-and-Algorithms---Workspace/tree/main/in-class-lab9>

## Section 1 : Implementing basic hash table

main.cpp	Output
<pre> 130     hashtable-&gt;insert(user_name, password); 131     break; 132     case 2: 133         /* delete item */ 134         cout &lt;&lt; "Enter item to be deleted: "; 135         cin &gt;&gt; user_name; 136         hashtable-&gt;delete_item(user_name); 137         break; 138     case 3: 139         /* hash lookup password*/ 140         cout &lt;&lt; "Enter user name to look up password: "; 141         cin &gt;&gt; user_name; 142         hashtable-&gt;hash_lookup(user_name); 143         break; 144     case 4: 145         hashtable-&gt;print_hashtable(); 146         break; 147     case -1: 148         /* hash lookup password*/ 149         cout &lt;&lt; "Exiting...\n"; 150         break; </pre>	<pre> /tmp/yxF8tcfzZq.o 0 Type command: 1 Enter user name: Lasitha Enter password to be saved: 200650 Password added! Type command: 4 [0]--&gt; [1]--&gt; [2]--&gt;200650 [3]--&gt; Type command: </pre>

main.cpp	Output
<pre> 81     } 82 } 83 void delete_item(string user_name) 84 { 85     int hash; 86     bool empty; 87     hash = hashfunc(user_name); 88     empty = is_slot_empty(hash); 89     if (empty) 90     { 91         cout &lt;&lt; "No item found\n"; 92     } 93     else 94     { 95         password[hash].clear(); 96         cout &lt;&lt; "User deleted\n"; 97     } 98 } 99 void print_hashtable() </pre>	<pre> Enter password to be saved: 200350 Slot occupied! Type command: 1 Enter user name: Chuti Enter password to be saved: 200250 Slot occupied! Type command: 4 [0]--&gt;200450 [1]--&gt;200550 [2]--&gt;200650 [3]--&gt; Type command: 1 Enter user name: Asna Enter password to be saved: 200222 Password added! Type command: 4 [0]--&gt;200450 [1]--&gt;200550 [2]--&gt;200650 [3]--&gt;200222 Type command: </pre>

Waiting for securepubads.g.doubleclick.net...

```

main.cpp
130     hashtable->insert(user_name, password);
131     break;
132     case 2:
133         /* delete item */
134         cout << "Enter item to be deleted: ";
135         cin >> user_name;
136         hashtable->delete_item(user_name);
137         break;
138     case 3:
139         /* hash lookup password*/
140         cout << "Enter user name to look up password:";
141         cin >> user_name;
142         hashtable->hash_lookup(user_name);
143         break;
144     case 4:
145         hashtable->print_hashtable();
146         break;
147     case -1:
148         /* hash lookup password*/
149         cout << "Exiting...\n";
150         break;

```

```

Output
[1]-->200550
[2]-->200650
[3]-->
Type command: 1
Enter user name: Asna
Enter password to be saved: 200222
Password added!
Type command: 4
[0]-->200450
[1]-->200550
[2]-->200650
[3]-->200222
Type command: 2
Enter item to be deleted: Lasitha
User deleted
Type command: 4
[0]-->200450
[1]-->200550
[2]-->
[3]-->200222
Type command:

```

The issue is when we try to add some users, we cannot add some of them. The root cause of this problem is that the hash function being used to determine the index for a particular value can produce identical indices for different values, which is known as a hash collision. When this occurs, two keys with the same hash values cannot be stored in the same index slot. To resolve this issue, various techniques like chaining or open addressing can be employed to manage these collisions.

## Section 2 : Implementing hash table with chaining

```

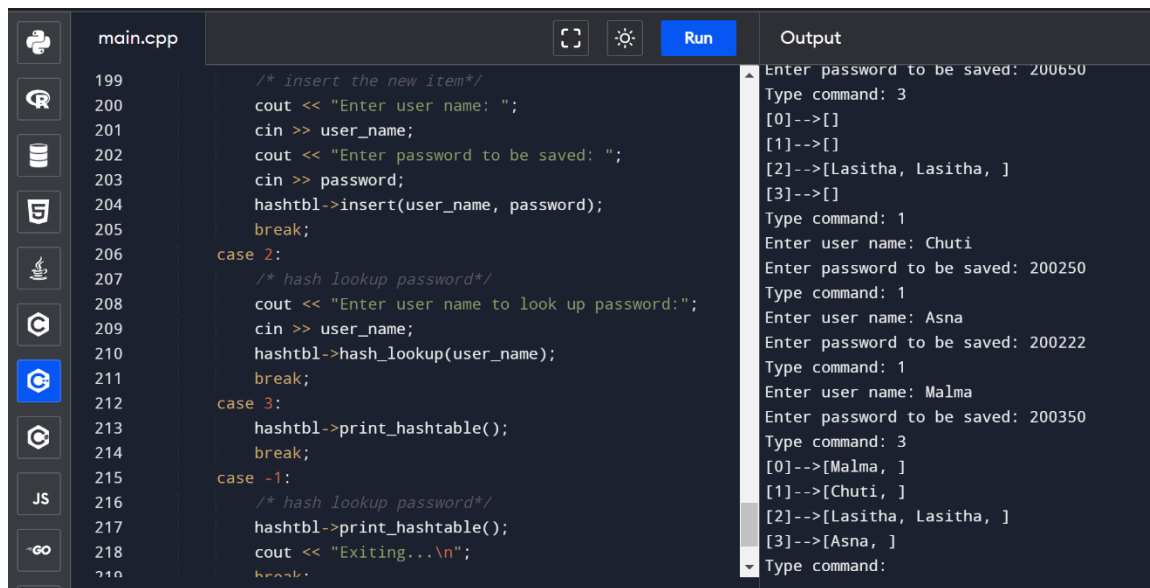
main.cpp
199     /* insert the new item*/
200     cout << "Enter user name: ";
201     cin >> user_name;
202     cout << "Enter password to be saved: ";
203     cin >> password;
204     hashtable->insert(user_name, password);
205     break;
206     case 2:
207         /* hash lookup password*/
208         cout << "Enter user name to look up password:";
209         cin >> user_name;
210         hashtable->hash_lookup(user_name);
211         break;
212     case 3:
213         hashtable->print_hashtable();
214         break;
215     case -1:
216         /* hash lookup password*/
217         hashtable->print_hashtable();
218         cout << "Exiting...\n";
219         break;

```

```

Output
/tmp/kOGVQqfmKP.o
0
Type command: 1
Enter user name: Lasitha
Enter password to be saved: 200650
Type command: 3
[0]-->[]
[1]-->[]
[2]-->[Lasitha, ]
[3]-->[]
Type command: 1
Enter user name: Lasitha
Enter password to be saved: 200650
Type command: 3
[0]-->[]
[1]-->[]
[2]-->[Lasitha, Lasitha, ]
[3]-->[]
Type command:

```



The screenshot shows a C++ IDE with a file named `main.cpp`. The code implements a hash table using an array of pointers to `Node` structures. It includes functions for inserting, looking up, and printing the hash table. The `main` function demonstrates these operations with a series of user inputs and commands. The output window on the right shows the execution results, including the insertion of three items and the subsequent lookups.

```
199      /* insert the new item*/
200      cout << "Enter user name: ";
201      cin >> user_name;
202      cout << "Enter password to be saved: ";
203      cin >> password;
204      hashtable->insert(user_name, password);
205      break;
206  case 2:
207      /* hash lookup password*/
208      cout << "Enter user name to look up password:";
209      cin >> user_name;
210      hashtable->hash_lookup(user_name);
211      break;
212  case 3:
213      hashtable->print_hashtable();
214      break;
215  case -1:
216      /* hash lookup password*/
217      hashtable->print_hashtable();
218      cout << "Exiting...\n";
219      break;
```

Output:

```
Enter password to be saved: 200650
Type command: 3
[0]-->[]
[1]-->[]
[2]-->[Lasitha, Lasitha, ]
[3]-->[]
Type command: 1
Enter user name: Chuti
Enter password to be saved: 200250
Type command: 1
Enter user name: Asna
Enter password to be saved: 200222
Type command: 1
Enter user name: Malma
Enter password to be saved: 200350
Type command: 3
[0]-->[Malma, ]
[1]-->[Chuti, ]
[2]-->[Lasitha, Lasitha, ]
[3]-->[Asna, ]
Type command:
```