CS2023 - Inclass Lab

Week 9 - Hash tables

Submission by: Sajeev Kugarajah (210554M)

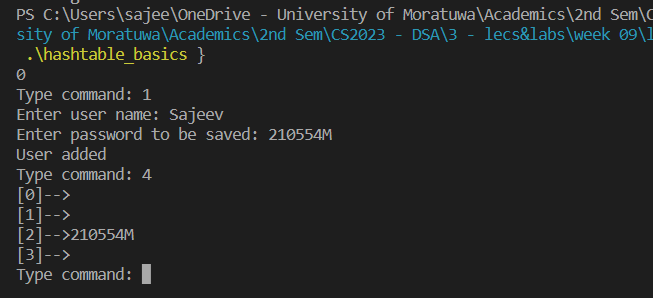
**Section 1 : Implementing basic hash table**

Expected submission

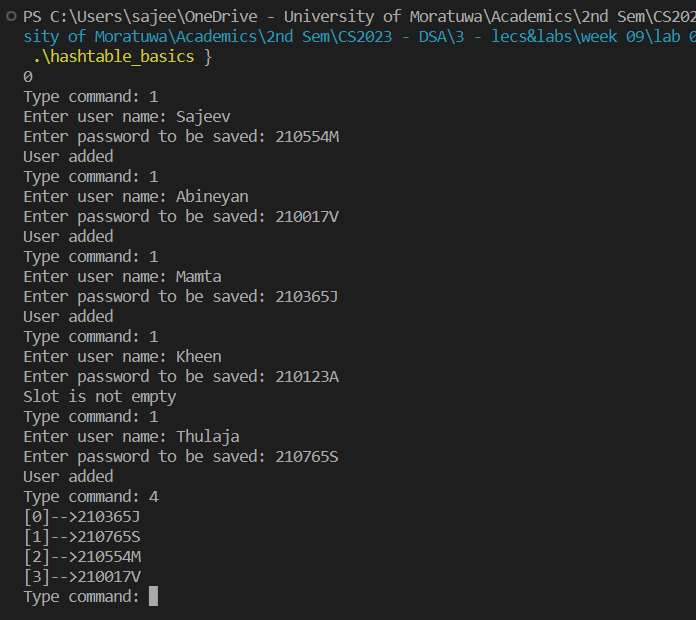
1. Complete *hashfunc, insert, hash lookup*

completed code file has been uploaded to the GitHub repository.

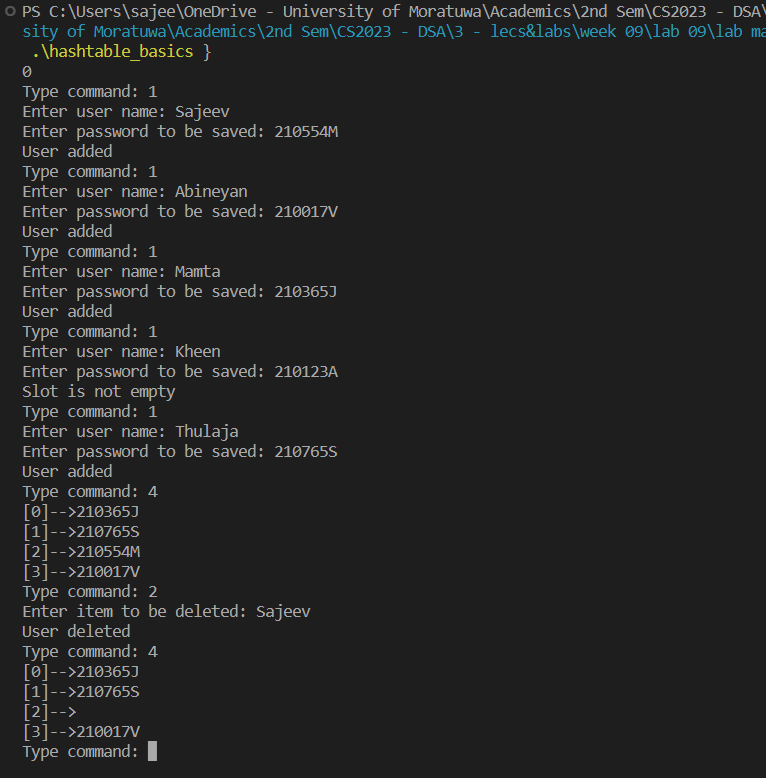
1. Insert your name as user name and your index number as password. Print hash table and take screenshot.



1. Add 3 more user names and passwords. Print hash table and take screenshot.



1. Delete your user name. Print hash table and take screenshot



1. What is issue when using a simple hash table like this and how can we change it, explain your answer.

Having a potential for collisions is the major issue in using simple hash tables like this. Collision is when more than one hash keys are assigned to same index. Hash functions we use, load factor of the hash table are the major factors which affect the probability of collision. And a hash table must have the ability to resolve the issue when collision occurs.

To overcome this issues we can use several techniques such as separate chaining, open addressing and more.

For example let’s talk about separate chaining.

Chaining is a technique where each key-value pairs of a hash table is stored as a linked list. When collision occurs with same hash codes, simply a new lined list created and appended to the existing array. Chaining is effective technique but the performance will be affected as the length of the linked list grows.

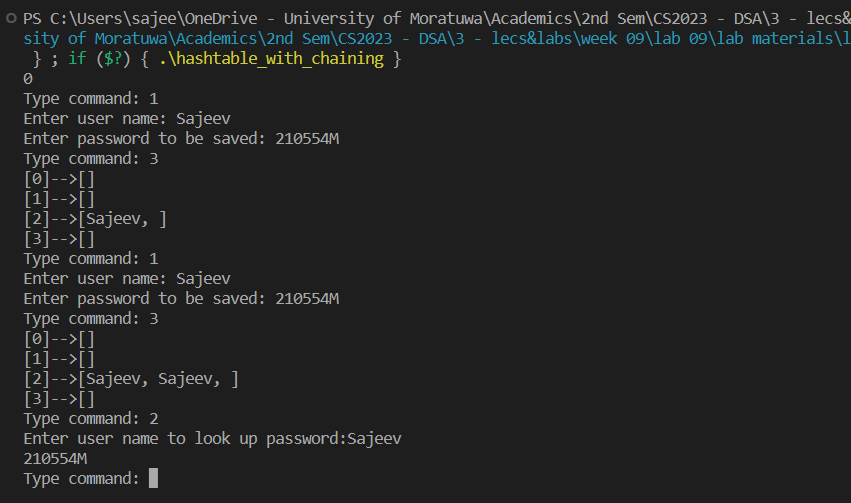
**Section 2 : Implementing hash table with chaining**

Expected submission

1. Complete *hash lookup*

completed code file has been uploaded to the GitHub repository.

1. Insert your name as user name and your index number as password(do it two times). Print hash table and take screenshot.



1. Add 3 more user names and passwords. Print hash table and take screenshot.

