CSE 4094

Special Topics in Computer Engineering
Advanced Data Structures

Project #2 Report
Visualization of Cuckoo Hashing

Ahmet Hamza Demir 150116025
Berk Yıldız 150117052

ABSTRACT

In our program, the number of tables (max 5 tables) and the size of tables (min 10 / max 30 cells) will be taken as input from the user. Also, the keys to be inserted, searched, and deleted can be taken from the user as in the example.

We implement this project in Java language.

FUNCTIONS

`CuckooHashTable`

It defines the class and creates an instance of the Cuckoo Hash Table. It initializes the number of tables, the size of the tables, and the table itself. It also prompts the user to input the number of tables to be created and the size of each table, with constraints on the minimum and maximum size. The function then provides a menu for the user to insert, search, and delete keys, as well as an option to exit the program. It also calls other functions within the class to perform these operations and keep track of statistics such as the number of collisions and the load factor of the table.

`hashFunction`

It generates an index for a given key where the key can be stored in the table. This function takes an integer key as an input, and returns an integer value representing the index. The hash function used is the modulus of the key by the size of the first table. This means that the index generated for a key is the remainder of the key divided by the size of the first table. The generated index will be within the range of 0 to the size of table minus one. So the function returns the position of the table where the key will be stored.

`insertKey`

It inserts a new key into the Cuckoo Hash Table. Users enter an input key to be inserted. Firstly it prints current versions of the tables and then uses the hashFunction to find the index for the key in the table. Then if the index is empty, the key is inserted into that position and the table number,index and final versions of the tables are displayed. If the index is not empty, then it increases the collision count, rehashes. Then it also displays the number of collisions ,the load factor of the table and modified versions of tables. It calculates the load factor by dividing the number of items inserted by tableSizes[tableNum] and multiplying by 100 to get the percentage.

`searchKey`

It searches for a key in the Cuckoo Hash Table. User enters an input key to be searched and then uses the hashFunction to find the index for the key in the table. It then iterates over the tables and checks if the key is present at the calculated index. If the key is found, it displays the table number and index where the key is located. If the key is not found, it displays that the key is not present in the table.

`deleteKey`

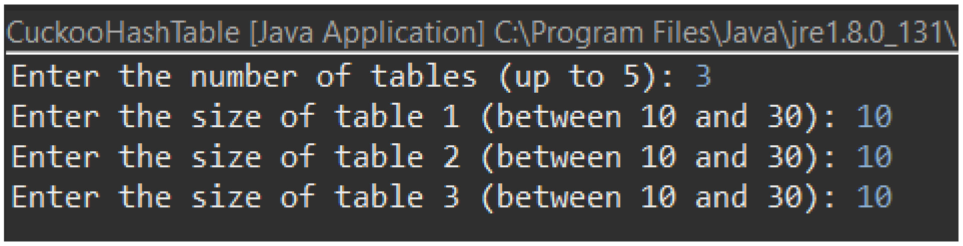
It deletes a key from the Cuckoo Hash Table. User enters an input key to be deleted. Firstly it prints current versions of the tables and then uses the hashFunction to find the index for the key in the table. It then iterates over the tables and checks if the key is present at the calculated index. If the key is found, it removes the key from that position and displays the table number and index where the key was located and that the key has been deleted. If the key is not found, it displays that the key is not present in the table. At last it shows modified versions of tables.

`visualization`

The purpose of the visualization() function is to display the contents of all tables. This function iterates over all tables and all elements of each table and prints the element.

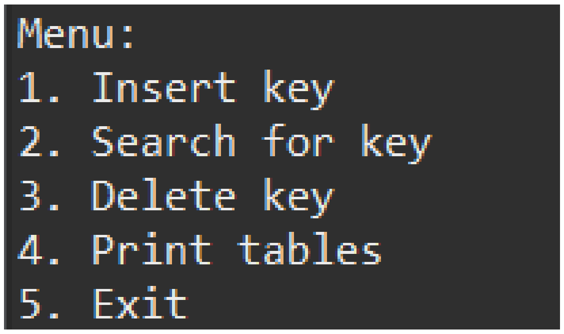
EXAMPLE OUTPUT

Creating Tables:



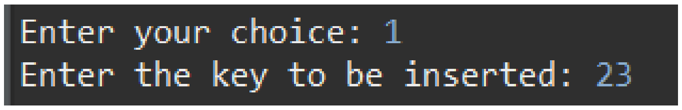
```
CuckooHashTable [Java Application] C:\Program Files\Java\jre1.8.0_131\
Enter the number of tables (up to 5): 3
Enter the size of table 1 (between 10 and 30): 10
Enter the size of table 2 (between 10 and 30): 10
Enter the size of table 3 (between 10 and 30): 10
```

Menu:



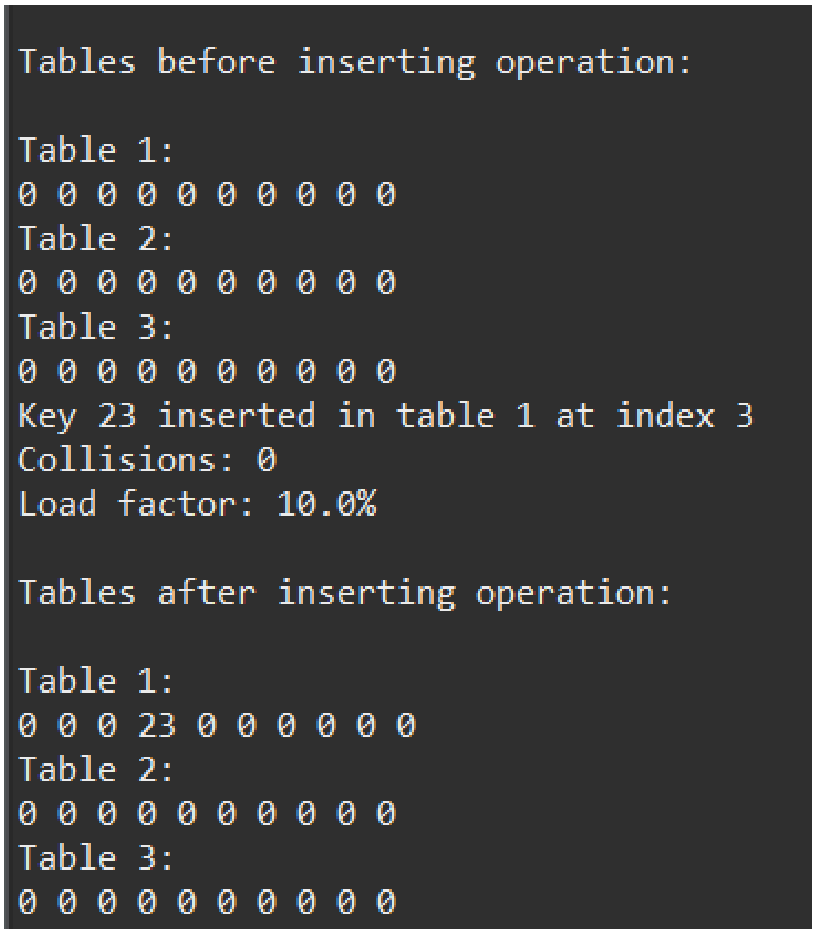
```
Menu:
1. Insert key
2. Search for key
3. Delete key
4. Print tables
5. Exit
```

Taking Choice from User:



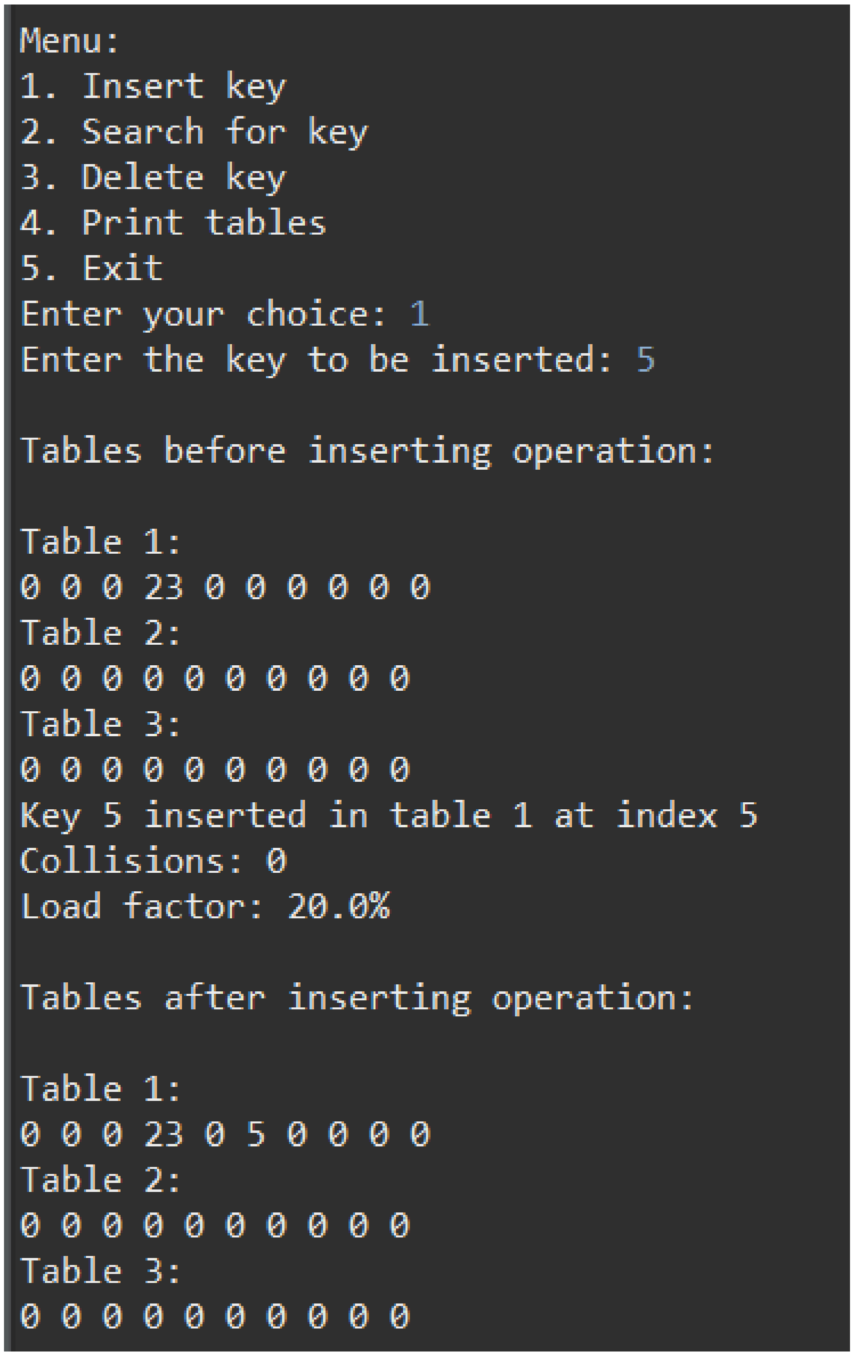
```
Enter your choice: 1
Enter the key to be inserted: 23
```

Inserting Operation:



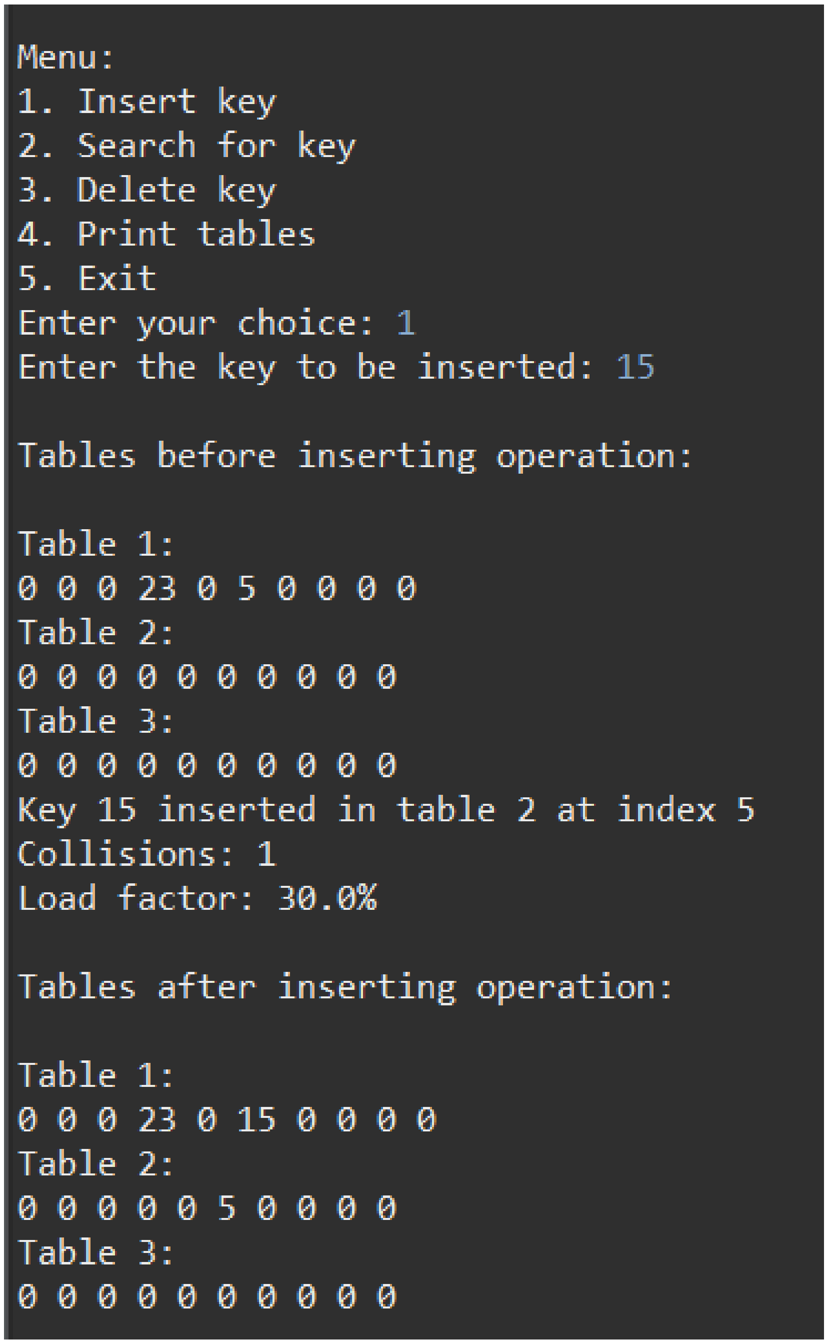
```
Tables before inserting operation:

Table 1:
0 0 0 0 0 0 0 0 0 0
Table 2:
0 0 0 0 0 0 0 0 0 0
Table 3:
0 0 0 0 0 0 0 0 0 0
Key 23 inserted in table 1 at index 3
Collisions: 0
Load factor: 10.0%

Tables after inserting operation:

Table 1:
0 0 0 23 0 0 0 0 0 0
Table 2:
0 0 0 0 0 0 0 0 0 0
Table 3:
0 0 0 0 0 0 0 0 0 0
```

Another Inserting Operation:



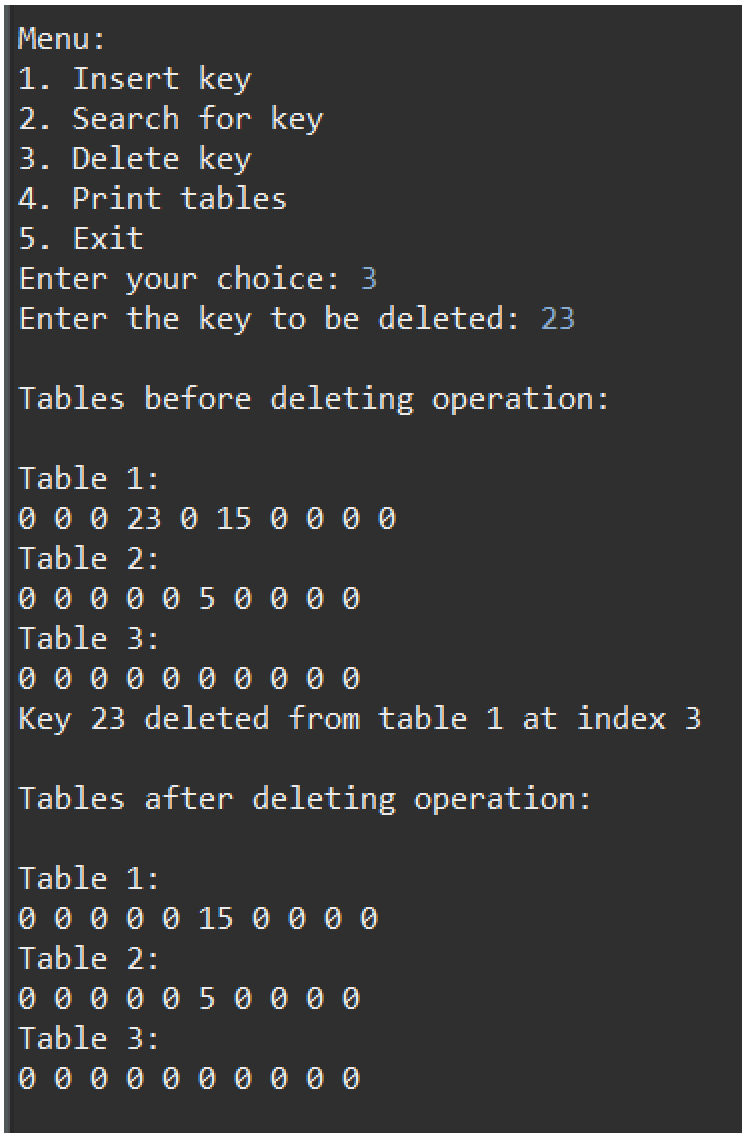
```
Menu:
1. Insert key
2. Search for key
3. Delete key
4. Print tables
5. Exit
Enter your choice: 1
Enter the key to be inserted: 5

Tables before inserting operation:

Table 1:
0 0 0 23 0 0 0 0 0 0
Table 2:
0 0 0 0 0 0 0 0 0 0
Table 3:
0 0 0 0 0 0 0 0 0 0
Key 5 inserted in table 1 at index 5
Collisions: 0
Load factor: 20.0%

Tables after inserting operation:

Table 1:
0 0 0 23 0 5 0 0 0 0
Table 2:
0 0 0 0 0 0 0 0 0 0
Table 3:
0 0 0 0 0 0 0 0 0 0
```

Inserting Operation with Collision:



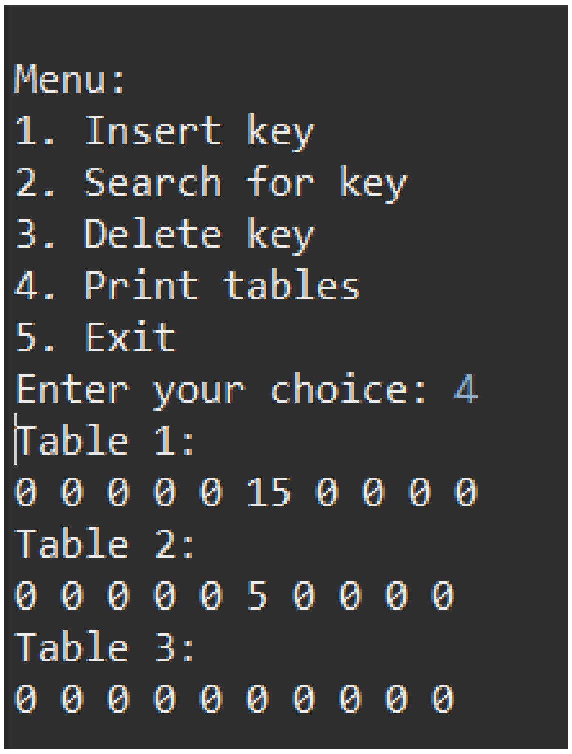
```
Menu:
1. Insert key
2. Search for key
3. Delete key
4. Print tables
5. Exit
Enter your choice: 1
Enter the key to be inserted: 15

Tables before inserting operation:

Table 1:
0 0 0 23 0 5 0 0 0 0
Table 2:
0 0 0 0 0 0 0 0 0 0
Table 3:
0 0 0 0 0 0 0 0 0 0
Key 15 inserted in table 2 at index 5
Collisions: 1
Load factor: 30.0%

Tables after inserting operation:

Table 1:
0 0 0 23 0 15 0 0 0 0
Table 2:
0 0 0 0 0 5 0 0 0 0
Table 3:
0 0 0 0 0 0 0 0 0 0
```

Delete Operation:



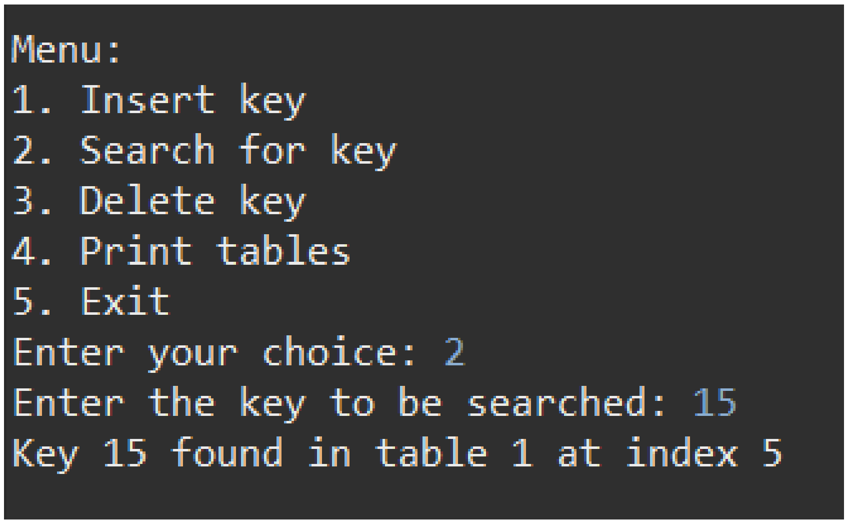
```
Menu:
1. Insert key
2. Search for key
3. Delete key
4. Print tables
5. Exit
Enter your choice: 3
Enter the key to be deleted: 23

Tables before deleting operation:

Table 1:
0 0 0 23 0 15 0 0 0 0
Table 2:
0 0 0 0 0 5 0 0 0 0
Table 3:
0 0 0 0 0 0 0 0 0 0
Key 23 deleted from table 1 at index 3

Tables after deleting operation:

Table 1:
0 0 0 0 0 15 0 0 0 0
Table 2:
0 0 0 0 0 5 0 0 0 0
Table 3:
0 0 0 0 0 0 0 0 0 0
```

Print Tables Operation:



```
Menu:
1. Insert key
2. Search for key
3. Delete key
4. Print tables
5. Exit
Enter your choice: 4
Table 1:
0 0 0 0 0 15 0 0 0 0
Table 2:
0 0 0 0 0 5 0 0 0 0
Table 3:
0 0 0 0 0 0 0 0 0 0
```

Search Operation:



```
Menu:
1. Insert key
2. Search for key
3. Delete key
4. Print tables
5. Exit
Enter your choice: 2
Enter the key to be searched: 15
Key 15 found in table 1 at index 5
```