# **Name: Abdurrahman Qureshi**

# **Roll No: 242466**

Practical No: 10

**1)** **Demonstrate the** Implementation of Hashing Function with different collision resolution technique.

CODE

#include <stdio.h>

#include <stdlib.h>

*struct* set

{

*int* key;

*int* data;

};

*struct* set \*array;

*int* capacity = 10;

*int* size = 0;

*int* i;

*int* hashFunction(*int* *key*)

{

    return (*key* % capacity);

}

*int* checkPrime(*int* *n*)

{

*int* i;

    if (*n* == 1 || *n* == 0)

    {

        return 0;

    }

    for (i = 2; i < *n* / 2; i++)

    {

        if (*n* % i == 0)

        {

            return 0;

        }

    }

    return 1;

}

*int* getPrime(*int* *n*)

{

    if (*n* % 2 == 0)

    {

*n*++;

    }

    while (!checkPrime(*n*))

    {

*n* += 2;

    }

    return *n*;

}

*void* init\_array()

{

    capacity = getPrime(capacity);

    array = (*struct* set \*)malloc(capacity \* sizeof(*struct* set));

    for (i = 0; i < capacity; i++)

    {

        array[i].key = 0;

        array[i].data = 0;

    }

}

*void* insert(*int* *key*, *int* *data*)

{

*int* index = hashFunction(*key*);

    if (array[index].data == 0)

    {

        array[index].key = *key*;

        array[index].data = *data*;

        size++;

        printf("\n Key (%d) has been inserted \n", *key*);

    }

    else if (array[index].key == *key*)

    {

        array[index].data = *data*;

    }

    else

    {

        printf("\n Collision occured  \n");

    }

}

*void* remove\_element(*int* *key*)

{

*int* index = hashFunction(*key*);

    if (array[index].data == 0)

    {

        printf("\n This key does not exist \n");

    }

    else

    {

        array[index].key = 0;

        array[index].data = 0;

        size--;

        printf("\n Key (%d) has been removed \n", *key*);

    }

}

*void* display()

{

*int* i;

    for (i = 0; i < capacity; i++)

    {

        if (array[i].data == 0)

        {

            printf("\n array[%d]: / ", i);

        }

        else

        {

            printf("\n key: %d array[%d]: %d \t", array[i].key, i, array[i].data);

        }

    }

}

*int* size\_of\_hashtable()

{

    return size;

}

*int* main()

{

*int* choice, key, data, n;

*int* c = 0;

    init\_array();

    do

    {

        printf("1.Insert item in the Hash Table"

               "\n2.Remove item from the Hash Table"

               "\n3.Check the size of Hash Table"

               "\n4.Display a Hash Table"

               "\n\n Please enter your choice: ");

        scanf("%d", &choice);

        switch (choice)

        {

        case 1:

            printf("Enter key -:\t");

            scanf("%d", &key);

            printf("Enter data -:\t");

            scanf("%d", &data);

            insert(key, data);

            break;

        case 2:

            printf("Enter the key to delete-:");

            scanf("%d", &key);

            remove\_element(key);

            break;

        case 3:

            n = size\_of\_hashtable();

            printf("Size of Hash Table is-:%d\n", n);

            break;

        case 4:

            display();

            break;

        default:

            printf("Invalid Input\n");

        }

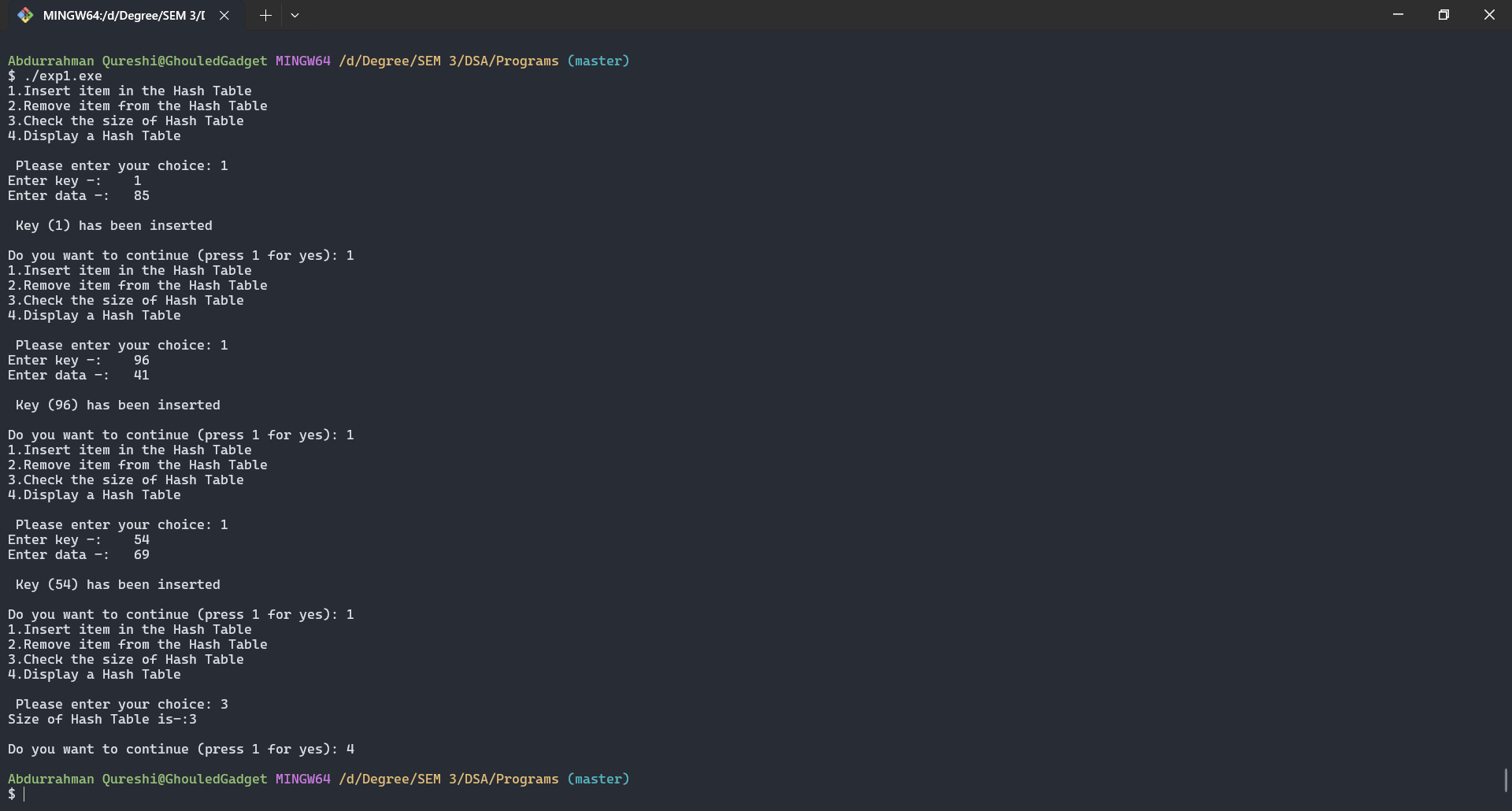
        printf("\nDo you want to continue (press 1 for yes): ");

        scanf("%d", &c);

    } while (c == 1);

}

OUTPUT



Tools used :

Software: Dev c++

Hardware: Lab Computers

References: Mam notes.

Conclusion

* Hash function is a function that uses the constant-time operation to store and retrieve the value from the hash table, which is applied on the keys as integers and this is used as the address for values in the hash table.
* Hashing is one of the important techniques in terms of searching data provided with very efficient and quick methods using hash function and hash tables.
* Each element can be searched and placed using different hashing methods.
* This technique is very faster than any other data structure in terms of time coefficient.