**Ex. No: 11**

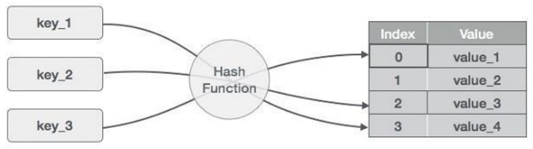
## HASHING TECHNIQUES

**AIM**: To write a C program to implement hash table

**DESCRIPTION:**

**Hashing**

Hashing is a technique to convert a range of key values into a range of indexes of an array. We're going to use modulo operator to get a range of key values. Consider an example of hash table of size 20, and the following items are to be stored. Item are in the (key,value) format.



- (1,20)
- (2,70)
- (42,80)
- (4,25)
- (12,44)
- (14,32)
- (17,11)
- (13,78)
- (37,98)

**ALGORITHM:**

1. Create a structure, data (hash table item) with key and value as data.
2. Now create an array of structure, data of some certain size (10, in this case). But, the size of array must be immediately updated to a prime number just greater than initial array capacity (i.e 10, in this case).
3. A menu is displayed on the screen.
4. User must choose one option from four choices given in the menu
5. Perform all the operations
6. Stop the program

**PROGRAM**

```
#include<stdio.h>

#include<stdlib.h>

struct data

{

      int

key;

      int

value;

}; struct data

*array; int

capacity = 10;

int size = 0;

/* this function gives a unique hash code to the given key

*/ int hashcode(int key)

{     return (key %

capacity);

}
```

```
/* it returns prime number just greater than array capacity

*/ int get_prime(int n)

{

                if (n % 2 == 0)

        {
```

```
n++;
```

```
                }
      for (; !if_prime(n); n += 2);
      return n;
}
/* to check if given input (i.e n) is prime or not
*/ int if_prime(int n)
{
                int i;
                if ( n == 1  ||  n == 0)
        {
                      return 0;
                }
                for (i = 2; i < n; i++)
        {
                      if (n % i == 0)
                {
                            return 0;}}
                return 1;
} void
init_array()
{
                int i;
      capacity = get_prime(capacity);     array = (struct
data*) malloc(capacity * sizeof(struct data));        for (i =
0; i < capacity; i++)
        {
            array[i].key = 0;
      array[i].value = 0;
```

```
                }}
/* to insert a key in the hash table */
void insert(int key)
```

```
{     int index =
hashcode(key);  if
(array[index].value == 0)
        {
            /*  key not present, insert it  */
            array[index].key = key;
      array[index].value = 1;
                      size++;
                      printf("\n Key (%d) has been inserted \n", key);
                }
                else if(array[index].key == key)
        {
            /*  updating already existing key  */     printf("\n Key (%d)
already present, hence updating its value \n", key);
                      array[index].value += 1;
                }
                else
        {
      /*  key cannot be insert as the index is already containing some other
key*/             printf("\n ELEMENT CANNOT BE INSERTED \n");
                }}
/* to remove a key from hash table */
void remove_element(int key)
{     int index  =
hashcode(key);
      if(array[index].value
== 0)
        {         printf("\n This key does not
exist \n");
```

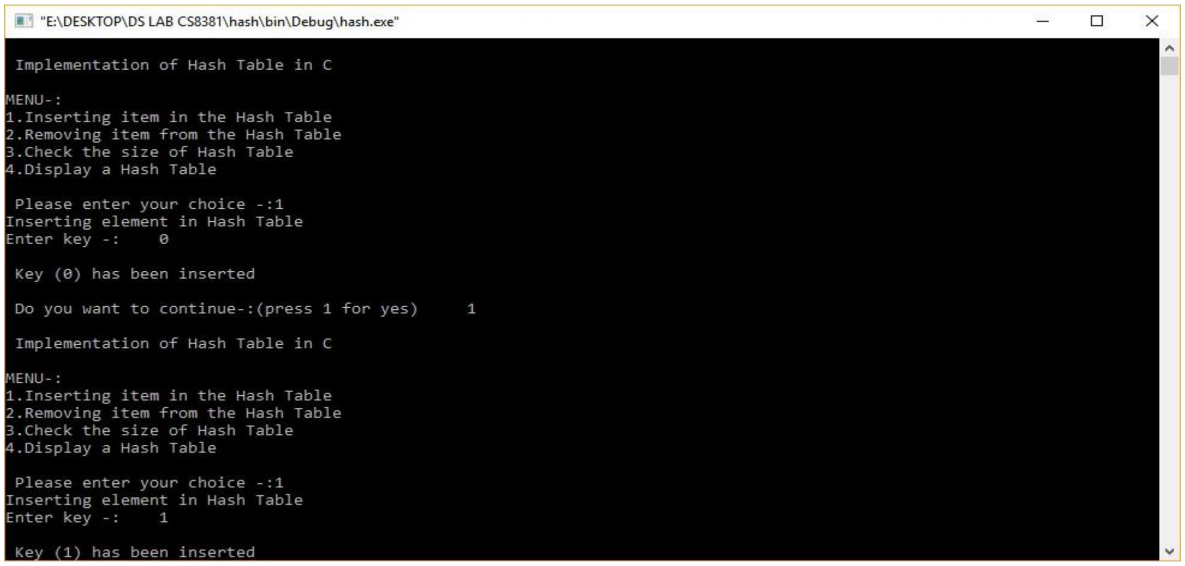
```
                }
      else {
      array[index].key = 0;
      array[index].value = 0;
                      size--;
                      printf("\n Key (%d) has been removed \n", key);}}
/* to display all the elements of a hash table */
void display()
```

```
{       int i;for (i = 0; i < capacity;
i++)

        {

                      if (array[i].value == 0)

                {

                            printf("\n Array[%d] has no elements \n");

                      }

                      else

                {

printf("\n array[%d] has elements -:\n key(%d) and value(%d) \t", i,
array[i].key, array[i].value);

                      }}}

int size_of_hashtable()

{

                return size;

} void

main()

{     int choice, key, value,

n, c;       init_array();

                do {

            printf("\n Implementation of Hash Table in C

\n\n");           printf("MENU-:  \n1.Inserting item in the

Hash Table"

                               "\n2.Removing 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("Inserting element in Hash

Table\n");                      printf("Enter key -:\t");

                   scanf("%d", &key);
```

```
insert(key);
```

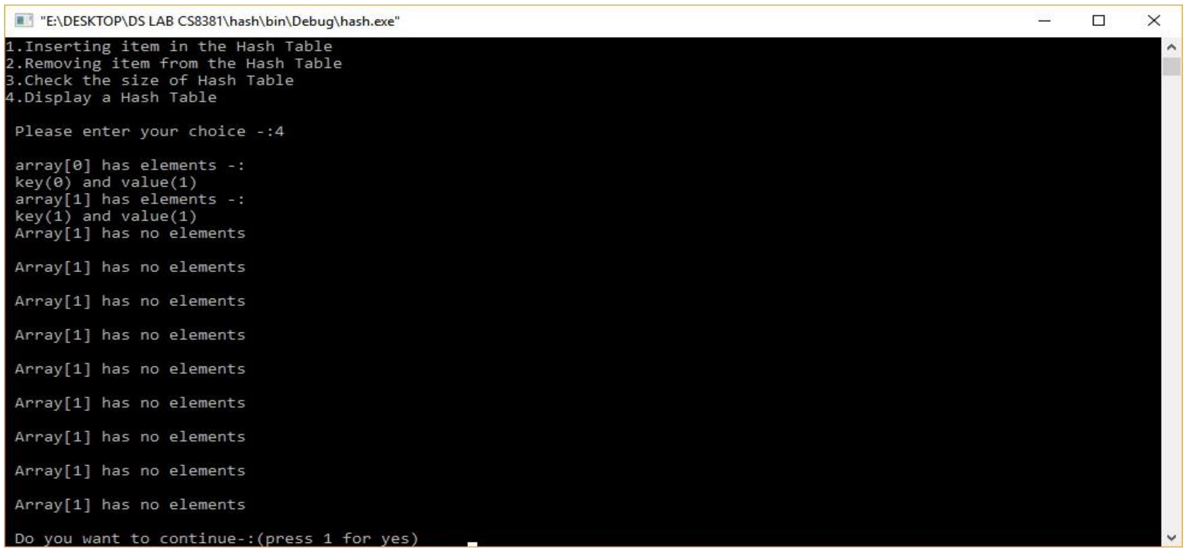
```
                  break;

            case 2:

                 printf("Deleting in Hash Table \n 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("Wrong Input\n");}          printf("\n

Do you want to continue-:(press 1 for yes)\t");

            scanf("%d",

&c);        }while(c ==

1);   getch();}
```

**<u>OUTPUT</u>**



"E:\DESKTOP\DS LAB CS8381\hash\bin\Debug\hash.exe"

```
 Implementation of Hash Table in C

MENU-:
1.Inserting item in the Hash Table
2.Removing item from the Hash Table
3.Check the size of Hash Table
4.Display a Hash Table

 Please enter your choice -:1
Inserting element in Hash Table
Enter key -:    0

 Key (0) has been inserted

 Do you want to continue-:(press 1 for yes)     1

 Implementation of Hash Table in C

MENU-:
1.Inserting item in the Hash Table
2.Removing item from the Hash Table
3.Check the size of Hash Table
4.Display a Hash Table

 Please enter your choice -:1
Inserting element in Hash Table
Enter key -:    1

 Key (1) has been inserted
```



"E:\DESKTOP\DS LAB CS8381\hash\bin\Debug\hash.exe"

```
1.Inserting item in the Hash Table
2.Removing item from the Hash Table
3.Check the size of Hash Table
4.Display a Hash Table

 Please enter your choice -:4

 array[0] has elements -:
 key(0) and value(1)
 array[1] has elements -:
 key(1) and value(1)
 Array[1] has no elements

 Array[1] has no elements

 Array[1] has no elements

 Array[1] has no elements

 Array[1] has no elements

 Array[1] has no elements

 Array[1] has no elements

 Array[1] has no elements

 Array[1] has no elements

 Do you want to continue-:(press 1 for yes)
```

**RESULT**: Thus the  C program to implemented Hash Table.