ASSIGNMENT 7

AIM:-

Insert the keys into into a hast table of length M using open addressing using double hashing with h(k)=1+(k*mod(m-1))

OBJECTIVE:-

To build a hash table using Double hashing in order to avoid collosion between 2 or more keys.

THEORY:-

Double hashing is a collision resolving technique in **Open Addressed** Hash tables. Double hashing uses the idea of applying a second hash function to key when a collision occurs.

Double hashing can be done using :
(hash1(key) + i * hash2(key)) % TABLE_SIZE
Here hash1() and hash2() are hash functions and TABLE_SIZE
is size of hash table.
(We repeat by increasing i when collision occurs)

First hash function is typically hash1(key) = key % TABLE_SIZE

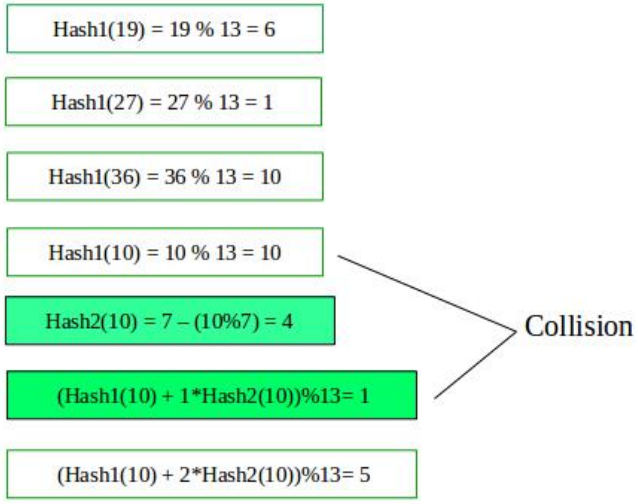
A popular second hash function is : **hash2(key) = PRIME – (key % PRIME)** where PRIME is a prime smaller than the TABLE_SIZE.

A good second Hash function is:

- It must never evaluate to zero
- Must make sure that all cells can be probed

Lets say, Hash1 (key) = key % 13

Hash2 (key) = 7 – (key % 7)



ALGORITHM:-

1.INSERT A KEY VALUE

```
void create(int table)
    {
        int i,key,index;
        string value;
        index=0;
       unsigned int index1=0;
            cout<<"enter a key value";
            cin>>key;
            cout<<"enter value";
            cin>>value;
            cout<<endl;
            index=key%table;
                if(a[index].key==0)
                {
                a[index].key=key;
                a[index].value=value;
                }
                else if(a[index].key!=0)
                {
                for(int j=1;j<table;j++)
                    {
                    index1=7-(key)%7;
                    index=(index+ j*index1)%10;
```

```
                    a[index].key=key;
                    a[index].value=value;
                    break;
                    }
                }
            }
```

2.DISPLAY THE TABLE

```
void display(int table)
{
    for(int i=0;i<table;i++)
    cout<<a[i].key<<a[i].value<<endl;
}
```

3.SEARCH A KEY-VALUE

```
void searching(int table)
{
    int count=0;
    int key;
    int i;
    cout<<"enter key-value to be searched"<<endl;
    cin>>key;
    for(i=0;i<table;i++)
    {
        if(a[i].key==key)
        {
            cout<<"search found"<<endl;
            cout<<"key"<<a[i].key<<endl;
            cout<<"value"<<a[i].value<<endl;
            break;
        }
        count++;
    }
    cout<<count;
}
```

CODE:-

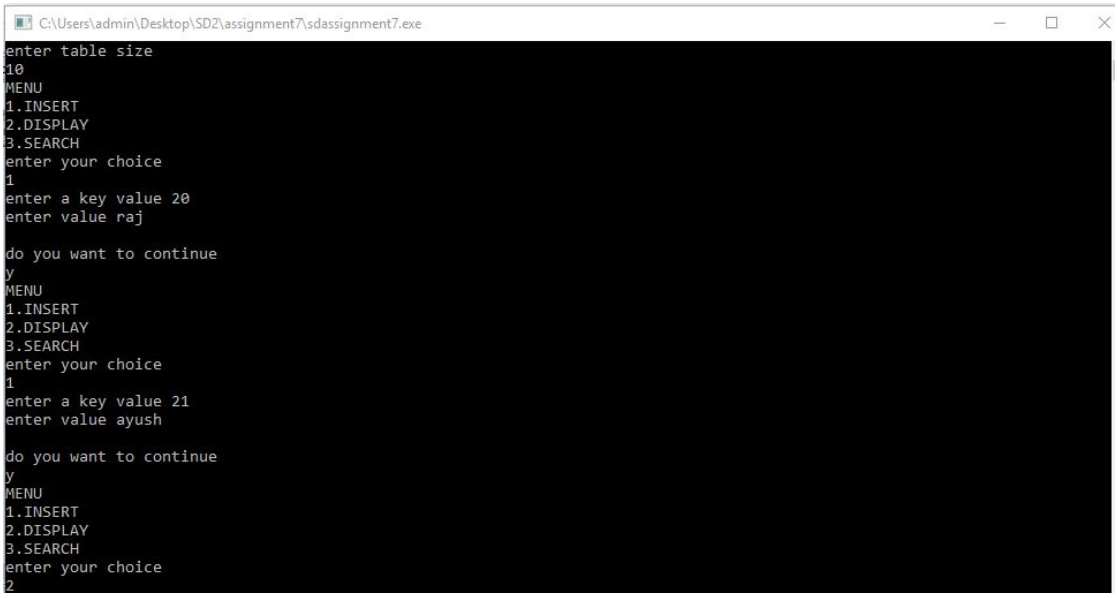
```
#include<iostream>
#define MAX 100
using namespace std;
class doublehashing
{
```

```
private:
    struct node
    {
        int key;
        string value;
    }a[MAX];
public:
    doublehashing()
    {
        for(int i=0;i<MAX;i++)
        {
        a[i].key=0;
        }
    }
    void create(int table)
    {
        int i,key,index;
        string value;
        index=0;
        unsigned int index1=0;
            cout<<"enter a key value";
            cin>>key;
            cout<<"enter value";
            cin>>value;
            cout<<endl;
            index=key%table;
                if(a[index].key==0)
                {
                a[index].key=key;
                a[index].value=value;
                }
                else if(a[index].key!=0)
                {
                for(int j=1;j<table;j++)
                    {
                    index1=7-(key)%7;
                    index=(index+ j*index1)%10;
                    a[index].key=key;
                    a[index].value=value;
                    break;
```

```
                        }
                    }
                }
    void display(int table)
    {
        for(int i=0;i<table;i++)
        cout<<a[i].key<<a[i].value<<endl;
    }
    void searching(int table)
    {
        int count=0;
        int key;
        int i;
        cout<<"enter key-value to be searched"<<endl;
        cin>>key;
        for(i=0;i<table;i++)
        {
            if(a[i].key==key)
            {
                cout<<"search found"<<endl;
                cout<<"key"<<a[i].key<<endl;
                cout<<"value"<<a[i].value<<endl;
                break;
            }
            count++;
        }
        cout<<count;
    }
};
int main()
{
    doublehashing l;
    char ch;
    int choice;
    int table;
    cout<<"enter table size"<<endl;
    cin>>table;
do{
   cout<<"MENU"<<endl;
   cout<<"1.INSERT"<<endl;
```

```
    cout<<"2.DISPLAY"<<endl;
    cout<<"3.SEARCH"<<endl;
    cout<<"enter your choice"<<endl;
    cin>>choice;
    switch(choice)
    {
    case 1:
            l.create(table);
            break;
    case 2:
        l.display(table);
        break;
    case 3:
        l.searching(table);
        break;
    }
    cout<<"do you want to continue"<<endl;
    cin>>ch;
}while(ch=='y');
}
```

OUTPUT:-



```
C:\Users\admin\Desktop\SD2\assignment7\sdassignment7.exe
3.SEARCH
enter your choice
2
20raj
21ayush
0
0
0
0
0
0
0
0
do you want to continue
y
MENU
1.INSERT
2.DISPLAY
3.SEARCH
enter your choice
3
enter key-value to be searched
20
search found
key20
valueraj
0do you want to continue
n

Process returned 0 (0x0)   execution time : 64.313 s
```

CONCLUSION:-

We have successfully implemented double hashing in hash table.