ASSIGNMENT 7

AIM: Insert the keys into a hash table of length m using open addressing using double hashing with h(k)=(1+kmod(m-1)).

OBJECTIVE: To study and learn the concepts of double hashing.

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.

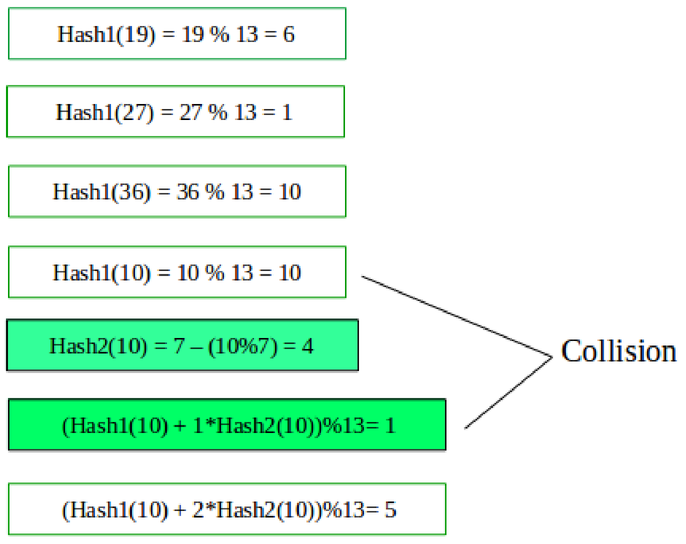
A good second Hash function is:

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

ALGORITHM:

Lets say, Hash1 (key) = key % 13

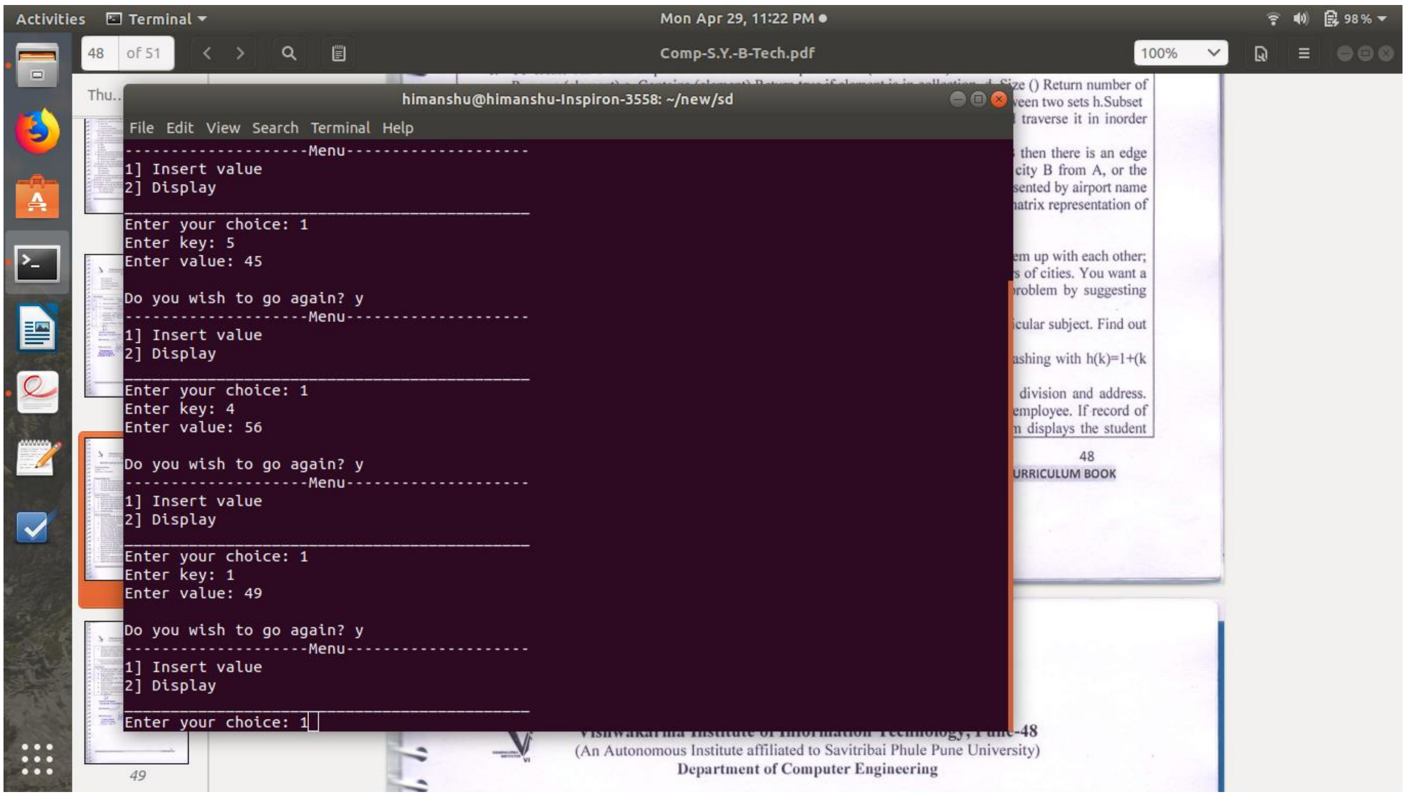
Hash2 (key) = 7 – (key % 7)



PROGRAM:

```
#include <iostream>
using namespace std;
class dr
{
   int n=10;
   int arr[100][3];
   int c;
public:
   dr()
   {
      cout<<"Table of size "<<n<<" created\n";
      for(int i=0;i<n;i++)
      {
         arr[i][0]=0;
         arr[i][1]=-1;
         arr[i][2]=-1;
      }
      c=0;
   }
   void add(int,int);
   int find_key(int);
   void display();
   void update_val(int,int);
};
void dr::add(int key,int value)
{
   int new_hash_addr1,new_hash_addr2,main_hash_addr=-1,j=0;
   if(this->find_key(key)!=-1)
   {
      cout<<"Key already exists\n";
      return;
   }
   if(c==(n-1))
   {
      cout<<"Table full, request denied\n";
   }
   new_hash_addr1=(key)%n;
   new_hash_addr1=1+(key%(n-1));
   if(arr[new_hash_addr1][1]==-1)
   {
      arr[new_hash_addr1][0]=key;
      arr[new_hash_addr1][1]=value;
   }
   else if(arr[new_hash_addr2][1]==-1)
   {
      arr[new_hash_addr2][0]=key;
      arr[new_hash_addr2][1]=value;
   }
   else
   {
      while(arr[new_hash_addr2][2]!=-1)
      {
         main_hash_addr=new_hash_addr2;
         new_hash_addr2=arr[main_hash_addr][2];
      }
      main_hash_addr=new_hash_addr2;
      for(int i=0;i<n;i++)
      {
         new_hash_addr2=(main_hash_addr+i)%n;
         if(arr[new_hash_addr2][1]==-1)
         {
            arr[new_hash_addr2][0]=key;
            arr[new_hash_addr2][1]=value;
            arr[main_hash_addr][2]=new_hash_addr2;
            c++;
            break;
         }
      }
   }
}
void dr::display()
{
   cout<<"Key\t\tValue\t\tChain\n";
   for(int i=0;i<n;i++)
   {
      cout<<arr[i][0]<<"\t\t"<<arr[i][1]<<"\t\t"<<arr[i][2]<<endl;
   }
}
int dr::find_key(int key)
{
   int search_addr=key%n,f=0;
   while(arr[search_addr][0]!=key && arr[search_addr][2]!=-1)
   {
      search_addr=arr[search_addr][2];
   }
   if(arr[search_addr][0]==key)
   {
      return arr[search_addr][1];
   }
   else if(arr[search_addr][2]==-1)
   {
      return -1;
   }
}
int main()
{
   char r;
   do
   {
      char op;
      dr table;
      int c;
      do
      {
         cout<<"--------------------Menu--------------------\n";
         cout<<"1] Insert value\n2] Display\n";
         cout<<"_______________________________________________\n";
         cout<<"Enter your choice: ";
         cin>>c;
         switch(c)
      {
        case 1: {
                    int key,val;
                    cout<<"Enter key: ";
                    cin>>key;
                    cout<<"Enter value: ";
                    cin>>val;
                    table.add(key,val);
                  }
                  break;
        case 2: table.display();
                  break;
        default:cout<<"Invalid\n";
      }
      cout<<"\nDo you wish to go again? ";
      cin>>op;
    }while(op=='y' || op=='Y');
    cout << "Test pass?(y/n): " << endl;
    cin>>r;
  }while(r=='n' || r=='N');
  cout<<"*****************\n";
  cout<<"*   Thank You!   *\n";
  cout<<"*****************\n";
  return 0;
}
```

OUTPUT:

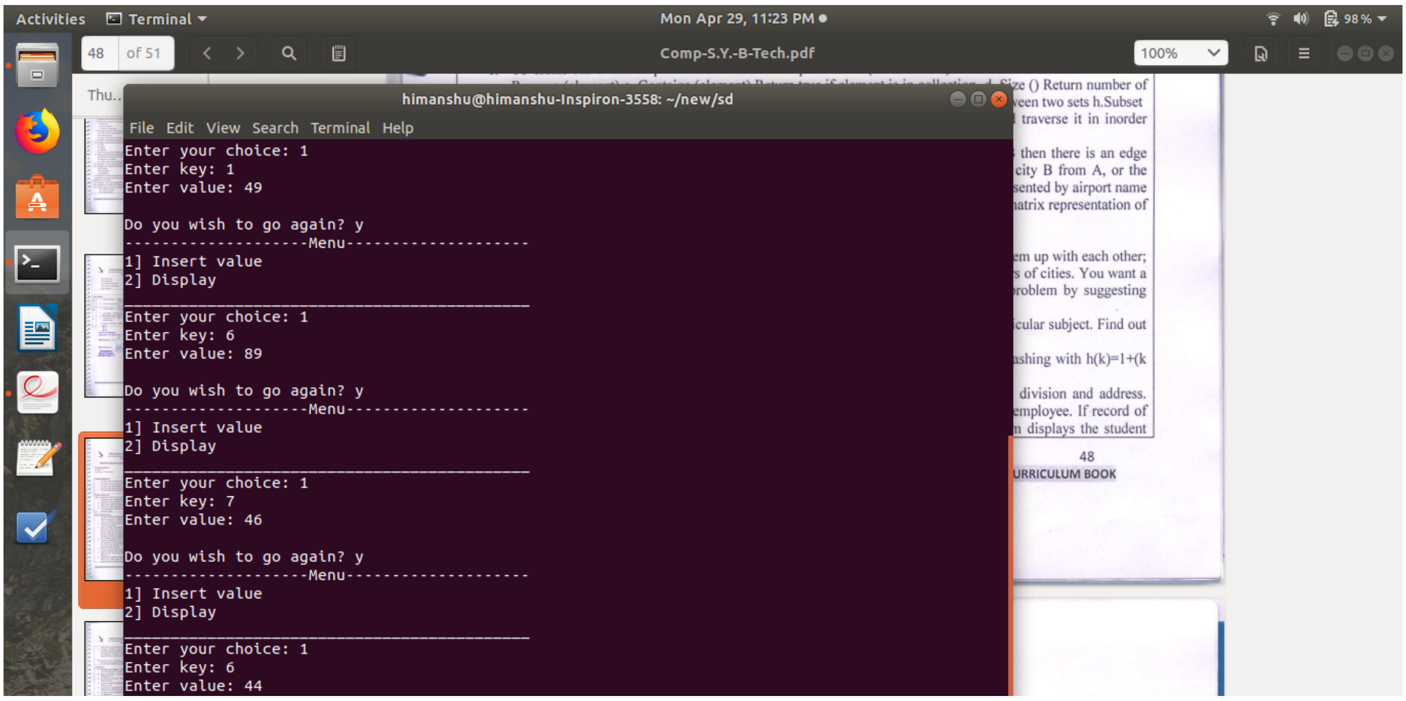


```
-------------------Menu--------------------
1] Insert value
2] Display
__________________________________________
Enter your choice: 1
Enter key: 5
Enter value: 45

Do you wish to go again? y
-------------------Menu--------------------
1] Insert value
2] Display
__________________________________________
Enter your choice: 1
Enter key: 4
Enter value: 56

Do you wish to go again? y
-------------------Menu--------------------
1] Insert value
2] Display
__________________________________________
Enter your choice: 1
Enter key: 1
Enter value: 49

Do you wish to go again? y
-------------------Menu--------------------
1] Insert value
2] Display
__________________________________________
Enter your choice: 1
```

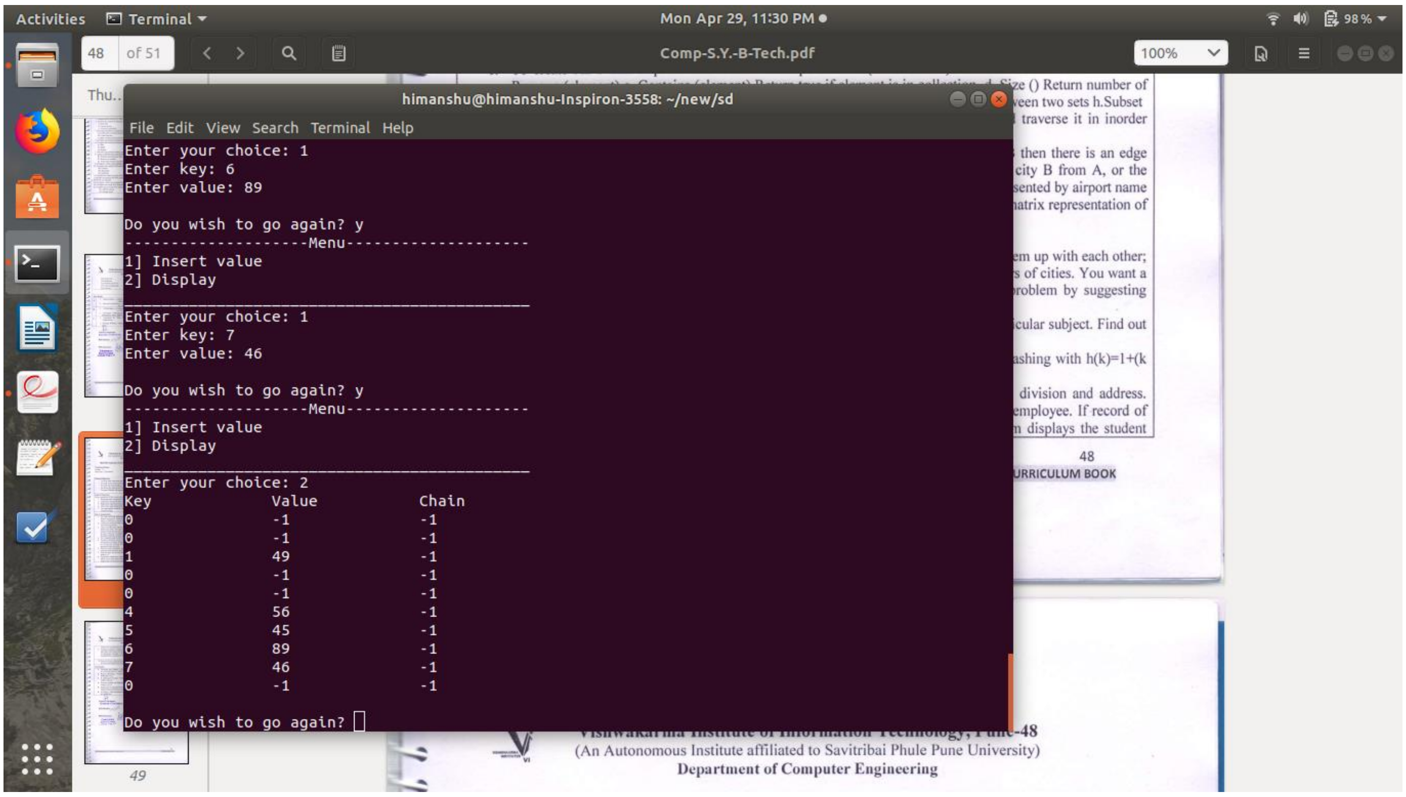


```
Enter your choice: 1
Enter key: 1
Enter value: 49

Do you wish to go again? y
-------------------Menu--------------------
1] Insert value
2] Display
__________________________________________
Enter your choice: 1
Enter key: 6
Enter value: 89

Do you wish to go again? y
-------------------Menu--------------------
1] Insert value
2] Display
__________________________________________
Enter your choice: 1
Enter key: 7
Enter value: 46

Do you wish to go again? y
-------------------Menu--------------------
1] Insert value
2] Display
__________________________________________
Enter your choice: 1
Enter key: 6
Enter value: 44
```


Activities
Terminal
Mon Apr 29, 11:30 PM
Comp-S.Y.-B-Tech.pdf
himanshu@himanshu-Inspiron-3558: ~/new/sd
File Edit View Search Terminal Help

```
Enter your choice: 1
Enter key: 6
Enter value: 89

Do you wish to go again? y
-------------------Menu-------------------
1] Insert value
2] Display
__________________________________________
Enter your choice: 1
Enter key: 7
Enter value: 46

Do you wish to go again? y
-------------------Menu-------------------
1] Insert value
2] Display
__________________________________________
Enter your choice: 2
Key             Value           Chain
0               -1              -1
0               -1              -1
1               49              -1
0               -1              -1
0               -1              -1
4               56              -1
5               45              -1
6               89              -1
7               46              -1
0               -1              -1

Do you wish to go again? 
```

CONCLUSION: We successfully implemented open addressing using double hashing.