#### **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

Lets say, Hash1 (key) = key % 13

(Hash1(10) + 2\*Hash2(10))%13= 5

Must make sure that all cells can be probed

#### **ALGORITHM:**

Hash1(19) = 19 % 13 = 6

Hash1(27) = 27 % 13 = 1

Hash1(36) = 36 % 13 = 10

Hash1(10) = 10 % 13 = 10

Hash2(10) = 
$$7 - (10\%7) = 4$$

Collision

(Hash1(10) + 1\*Hash2(10))%13= 1

Hash2 (key) = 7 - (key % 7)

### **PROGRAM:**

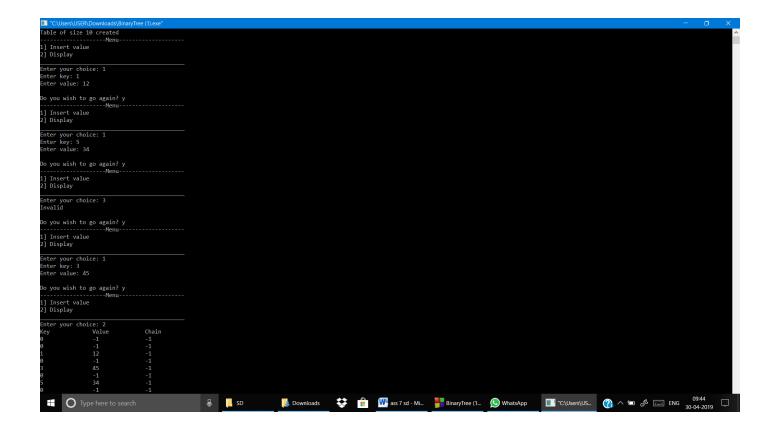
```
#include <iostream>
using namespace std;
class dr
  int n=10;
  int arr[100][3];
  int c;
public:
  dr()
    cout<<"Table of size "<<n<<" created\n";
    for(int i=0;i<n;i++)
      arr[i][0]=0;
      arr[i][1]=-1;
      arr[i][2]=-1;
    }
    c=0;
  void add(int,int);
  int find key(int);
  void 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";</pre>
  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<<"----\n";
    cout<<"1] Insert value\n2] Display\n";
                                                                \n";
    cout<<"Enter your choice: ";</pre>
    cin>>c;
    switch(c)
    {
      case 1: {
             int key,val;
             cout<<"Enter key: ";
             cin>>key;
             cout<<"Enter value: ";</pre>
             cin>>val;
             table.add(key,val);
          break;
      case 2: table.display();
          break;
      default:cout<<"Invalid\n";</pre>
    cout<<"\nDo you wish to go again? ";</pre>
    cin>>op;
  }while(op=='y' || op=='Y');
  cout << "Test pass?(y/n): " << endl;</pre>
  cin>>r;
}while(r=='n' || r=='N');
cout<<"****************\n";
cout<<"* Thank You! *\n";</pre>
cout<<"**************\n";
return 0;
```

# **OUTPUT:**

}



**CONCLUSION:** We successfully implemented open addressing using double hashing.