**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

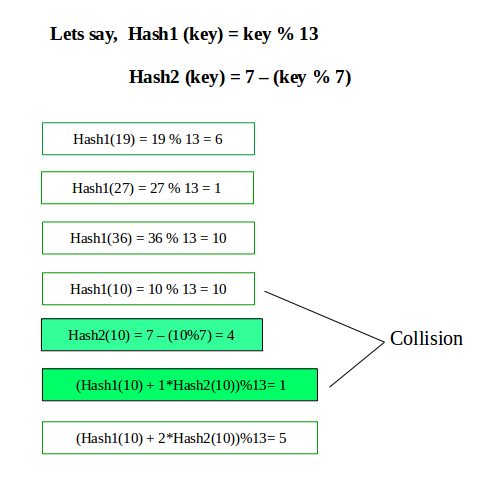
A popular second hash function is:

**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:**



**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";

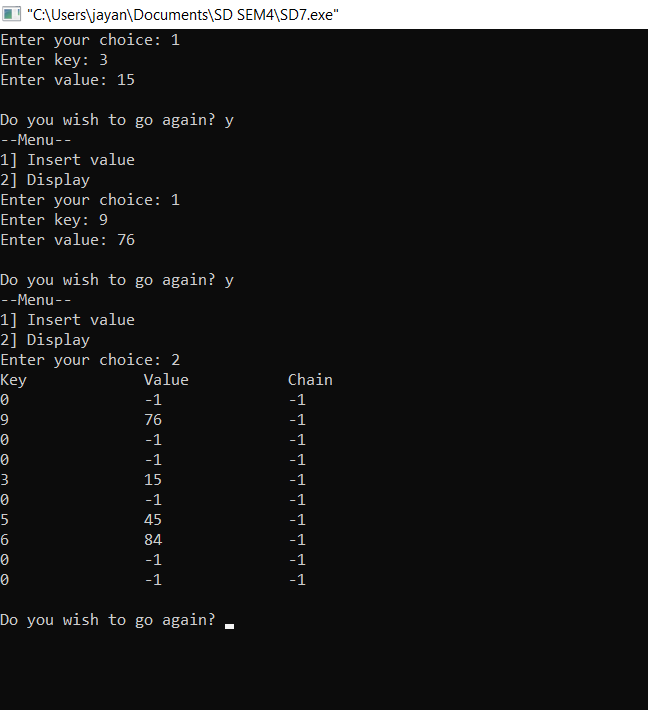
cout<<"\* Thank You! \*\n";

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

return 0;

}

**OUTPUT:**

****

**CONCLUSION:**

We successfully implemented open addressing using double hashing.