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)

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

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

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

Collision

PROGRAM :

```
#include <iostream>

using namespace std;

const int TABLE_SIZE = 10;
int hashTable[TABLE_SIZE] ={0};

void addInTable(){
    int key;
    bool isPlaced = false;
    cout<<"Enter the key to be inserted in the table : ";
    cin>>key;
    int Hash1 = key % TABLE_SIZE;
    int Hash2 = 7 - (key % 7);
    if(hashTable[Hash1] == 0){
```

```
        hashTable[Hash1] = key;
        isPlaced = true;
    }
    else if(hashTable[Hash2] == 0){
        hashTable[Hash2] = key;
        isPlaced = true;
    }
    else{
        for(int i = 0;i<TABLE_SIZE; i++){
            if(hashTable[Hash1 + (i*Hash2)] == 0){
                hashTable[Hash1 + (i*Hash2)] = key;
                isPlaced = true;
            }
        }
    }
    if(!isPlaced){
        cout<<"The number is not inserted as array is full."<<endl;
    }
}

void displayTable(){
    for(int i = 0; i<TABLE_SIZE; i++){
        cout<<hashTable[i]<<"  ";
    }
    cout<<endl;
}

int main()
{
    char ch;
    do{
        cout<<"::::::::::::::::::::::::::::::"<<endl;
```

```
        cout<<"1.add In hash Table"<<endl<<"2.show the table."<<endl;
        cout<<endl<<"Enter the choice : ";
        int choice;
        cin>>choice;
        switch(choice){
        case 1 :
            cout<<"How many elements do you want to add ? ";
            int no;
            cin>>no;
            while(no != 0){
                addInTable();
                no--;
            }
        break;
        case  2: displayTable();
        break;
        default: cout<<"Wrong Input !!"<<endl;
        }
        cout<<"Do you want to continue ? [Y/N] ";
        cin>>ch;
    }while(ch=='y' || ch=='Y');

    return 0;
}
```

OUTPUT:

"E:\codeblocksprogram\Sd- double hashing\bin\Debug\Sd- double hashing.exe"

```
:::::::::::::::::::::::::::::
1.add In hash Table
2.show the table.

Enter the choice : 1
How many elements do you want to add ? 7
Enter the key to be inserted in the table : 23
Enter the key to be inserted in the table : 33
Enter the key to be inserted in the table : 43
Enter the key to be inserted in the table : 89
Enter the key to be inserted in the table : 56
Enter the key to be inserted in the table : 55
Enter the key to be inserted in the table : 11
Do you want to continue ? [Y/N] y
:::::::::::::::::::::::::::::
1.add In hash Table
2.show the table.

Enter the choice : 2
0  11  33  23  0  55  43  56  0  89
Do you want to continue ? [Y/N] n

Process returned 0 (0x0)   execution time : 50.068 s
Press any key to continue.
```

CONCLUSION:

We successfully implemented open addressing using double hashing.