ASSIGNMENT : 7

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

Objective:-
to study the hashing concept and its techniques.

Theory:-

Hashing is an algorithm that calculates a fixed-size bit string value from a file. A file basically contains blocks of data. Hashing transforms this data into a far shorter fixed-length value or key which represents the original string. The hash value can be considered the distilled summary of everything within that file.

Hashing is implemented in two steps:

1. An element is converted into an integer by using a hash function. This element can be used as an index to store the original element, which falls into the hash table.
2. The element is stored in the hash table where it can be quickly retrieved using hashed key.

   hash = hashfunc(key)
   index = hash % array_size

a) Linear Probing: In linear probing, we linearly probe for next slot. For example, typical gap between two probes is 1 as taken in below example also.
let **hash(x)** be the slot index computed using hash function and **S** be the table size

Double Hashing:-

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.
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

Quadratic probing is similar to linear probing and the only difference is the interval between successive probes or entry slots. Here, when the slot at a hashed index for an entry record is already occupied, you must start traversing until you find an unoccupied slot. The interval between slots is computed by adding the successive value of an arbitrary polynomial in the original hashed index.

Algorithm:-

The primary hash function (that determining the bin) is the object statically cast as an int (see static_cast<int>) and taking this integer module *M* (i % M) and adding *M* if the value is negative. The secondary (odd) hash function (that determining the jump size) is **derived** from the integer divided by *M* modulo *M* ((i / M) % M) which is again made positive, if necessary, by adding *M*.

Program Code:-

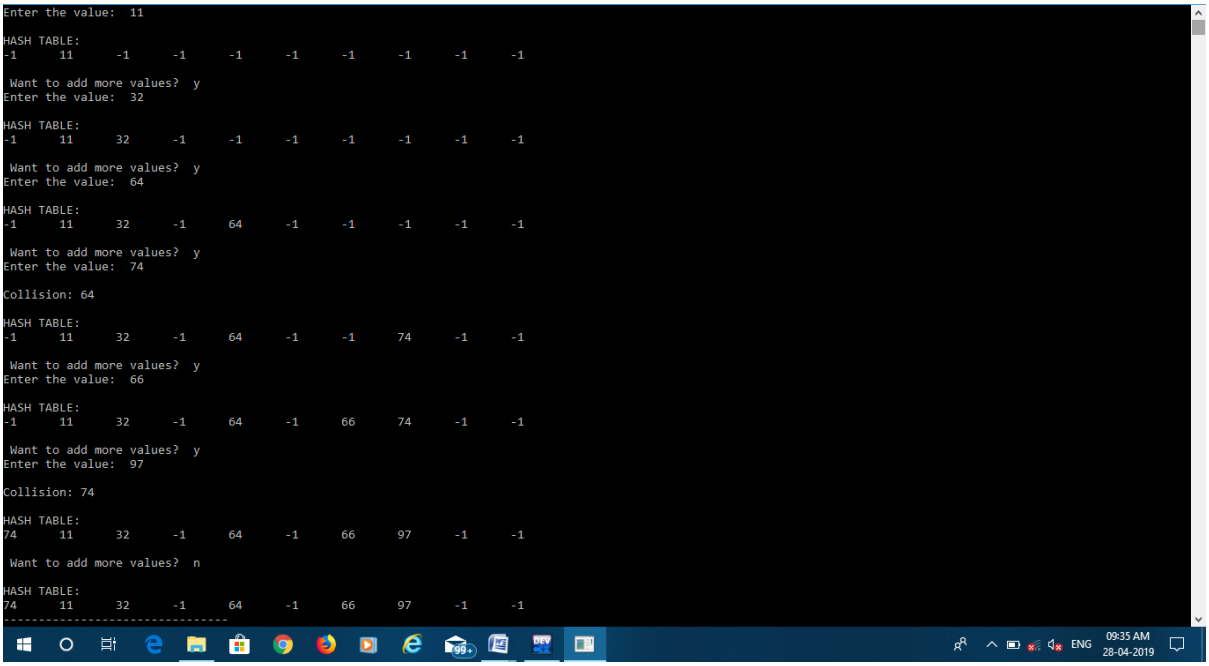
```
#include<iostream>

#include<stdlib.h>

using namespace std;
```

```
int size=10;
void display(int hash[])
{
	int i;
	cout<<"\nHASH TABLE:\n";
	for(i=0;i<size;i++)
	{
		cout<<""<<hash[i]<<"\t";
	}
}
int main()
{
	int hash[size],val,i;
	char ch;
	for(i=0;i<size;i++)
	{
		hash[i]=-1;
	}
	do
	{
		cout<<"Enter the value:  ";
		cin>>val;
```

```
            int m=val%size;
            if(hash[m] == -1)
            {
                  hash[m]=val;
                  display(hash);
            }
      else
      {
            cout<<"\nCollision: ";
            cout<<""<<hash[m]<<"\n";
            if(hash[m]%10!=m)
            {
                  int h=hash[m];
                  hash[m]=val;
                  val=h;
            }
            int x=1+(val%(size-1));
            for(i=1;i<size;i++)
            {
                  int y=(val+i*x);
                  int z=y%size;
                  if(hash[z]==-1)
                  {
```

```
                        hash[z]=val;
                        display(hash);
                        break;
                    }
                }
            }
            cout<<"\n\n Want to add more values?  ";
            cin>>ch;
            int ss=0;
            for(i=0;i<size;i++)
            {
                if(hash[i]!= -1)
                ss++;
            }
            if(ss==10)
            {
                cout<<"\n\nHast table is full";
                exit(1);
            }
}while(ch=='y' || ch=='Y');
display(hash);
return 0;
}
```

Output Screenshots:-



Conclusion:-

Thus,we have studied double hashing and hashing technique.