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

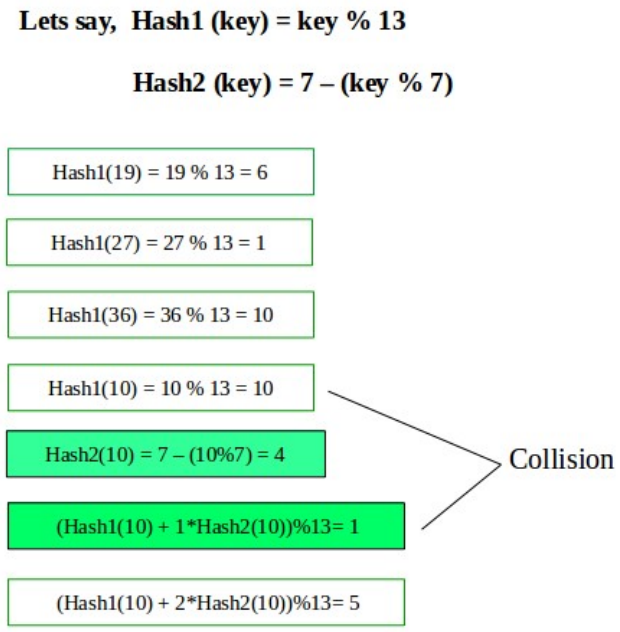
A popular second hash function is:

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:



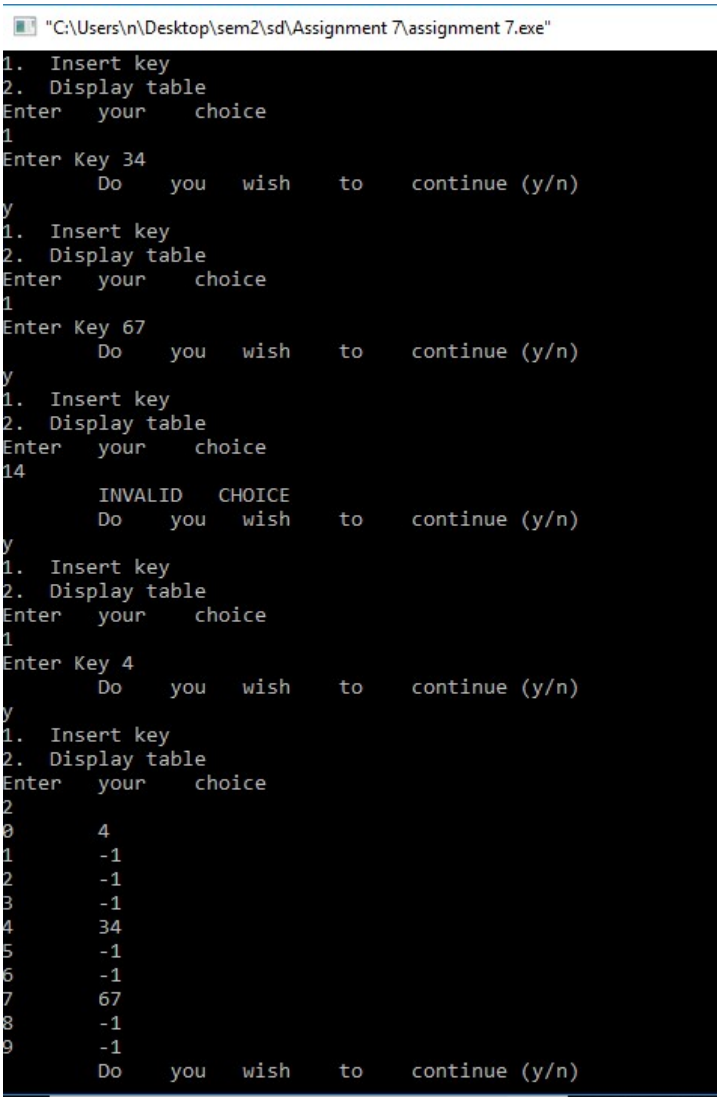
Program:

```
#include<iostream>
#include<string>
using namespace std;
class Hash
{
int k;
int ht[10];
public:
Hash()
{
for(int i=0; i<10;i++)
{
ht[i]= -1;
}
}
void insert()
{
int k;
 cout<<"Enter Key ";
 cin>>k;
int pos1, pos2, pos;
pos1 = (k) % 10;
if (ht[pos1] ==-1)
{
     ht[pos1] = k;
```

```
}
else
{
pos2=(7 - (k % 7));
int i = 1;
        while (1)
        {
            int pos = (pos1 + i * pos2) % 10;
            if (ht[pos] == -1)
            {
                ht[pos] = k;
                break;
            }
            i++;
        }
    }
}
void display()
{
for(int i=0; i<10; i++)
{
cout<<i << "\t" << ht[i] <<endl;
}
}
};
```

```
int main()
{
Hash h;
int ch;
char choice;
 do
  {
 cout  <<  "1.  Insert key "  <<  endl;
cout  <<  "2.  Display table "  <<  endl;
 cout  <<  "Enter  your  choice  "  <<  endl;
  cin  >>  ch  ;
 switch  (  ch  )
 {
 case  1:
  h.insert();
  break;
 case  2  :
h.display();
  break;
  default  :
  cout  <<  "\tINVALID  CHOICE  "  <<  endl;
 }
  cout  <<  "\tDo  you  wish  to  continue (y/n) "  <<  endl;
 cin  >>  choice  ;
  } while  (  choice  ==  'y');
}
```

Output:



Conclusion:

We successfully implemented open addressing using double hashing.

It is a better collision resolution technique compared to single hashing, since it has less number of collisions.