**Name : Resham Landge**

**Roll No : 2339**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***Assignment No : 9**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Title** **:** Store data of students with telephone no and name in the structure using hashing function for telephone number and implement chaining with and without replacement.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include<iostream>

#define MAX 10

using namespace std;

class Hash

{

public:

int table[MAX];

void linear\_without\_rep();

void linear\_with\_rep();

int hash(int key);

int empty(int table[MAX],int loc);

};

int Hash::hash(int key) //hash function for creating hash table

{

return(key % 10);

}

int Hash::empty(int table[MAX],int loc) //function to find empty fields in table

{

int i=loc;

do

{

i++;

i=i % MAX;

}while(table[i]!=-1 && 1!=loc); //continue the loop until i & loc is not same

return i;

}

void Hash::linear\_without\_rep() //linear probing without replacement

{

int key,loc,pos,i=0;

char ch;

for(i=0; i<MAX; i++)

table[i]=-1;

cout<<"\n\n\tHash Table\tHash Key";

for(i=0; i<MAX; i++)

cout<<"\n\t"<<i<<"\t\t"<<table[i];

i=0;

do

{

cout<<"\n\n\tEnter data : ";

cin>>key;

loc=hash(key); //call hash key function to find location

if(table[loc]==-1) //if loc is empty then copy to it

table[loc]=key;

else

{

pos=empty(table,loc); //if loc not empty then

if(pos!=loc) //find its next empty space in table

table[pos]=key;

else

cout<<"\n\tHash Table Full ";

}

cout<<"\n\tHash Table\tHash key";

for(i=0; i<MAX; i++)

cout<<"\n\t"<<i<<"\t\t"<<table[i];

cout<<"\n\n\tDo U want more element(y/n) : ";

cin>>ch;

}while(ch=='Y' || ch=='y');

}

void Hash::linear\_with\_rep() //linear probing with replacement

{

int key,i=0,loc,pos;

char ch;

for(i=0; i<MAX; i++)

table[i]=-1;

cout<<"\n\n\tHash Table\tHash Key";

for(i=0; i<MAX; i++)

cout<<"\n\t"<<i<<"\t\t"<<table[i];

i=0;

do

{

cout<<"\n\tEnter data : ";

cin>>key;

loc=hash(key); //call hash key function to find location

if(table[loc]==-1) //if location is empty then copy it

table[loc]=key;

else

{

pos=empty(table,loc); //if it contains data then it is null

if(pos==loc)

cout<<"\n\tHash table Full";

else

{

if(loc==hash(table[loc])) //replace that key

table[pos]=key;

else

{

table[pos]=table[loc];

table[loc]=key;

}

}

}

cout<<"\n\tHash table\tHash Key";

for(i=0; i<MAX; i++)

cout<<"\n\t"<<i<<"\t\t"<<table[i];

cout<<"\n\n\tDo U want more element(y/n) : ";

cin>>ch;

}while(ch=='Y' || ch=='y');

}

int main()

{

Hash h;

int choice;

char ch;

do

{

cout<<"\n\t1.Linear Probing Without Replacement\n\t2.Linear Probing with Replacement\n";

cout<<"\n\tEnter your choice : ";

cin>>choice;

switch(choice)

{

case 1:

h.linear\_without\_rep();

break;

case 2:

h.linear\_with\_rep();

break;

}

cout<<"\n\tDo U want to continue(y/n) : ";

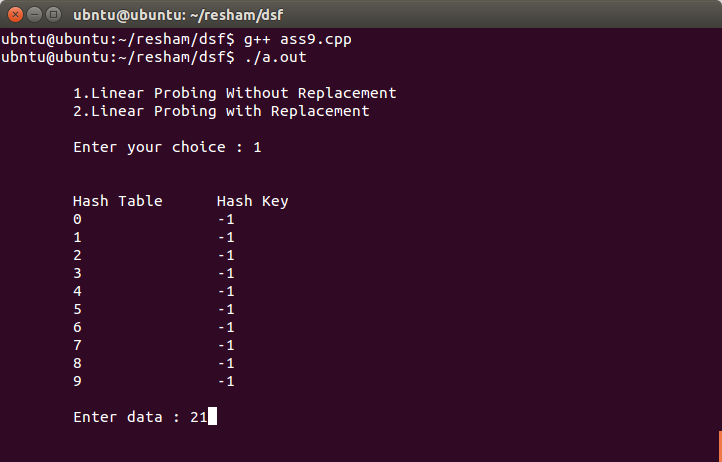
cin>>ch;

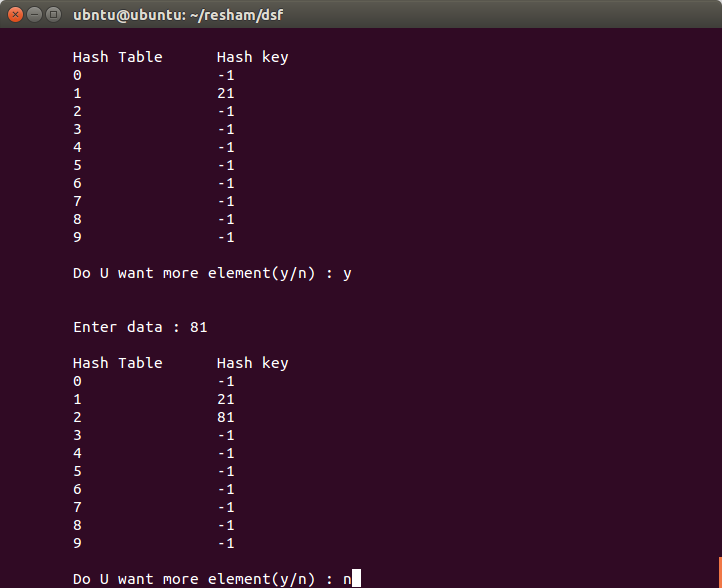
}while(ch=='Y'||ch=='y');

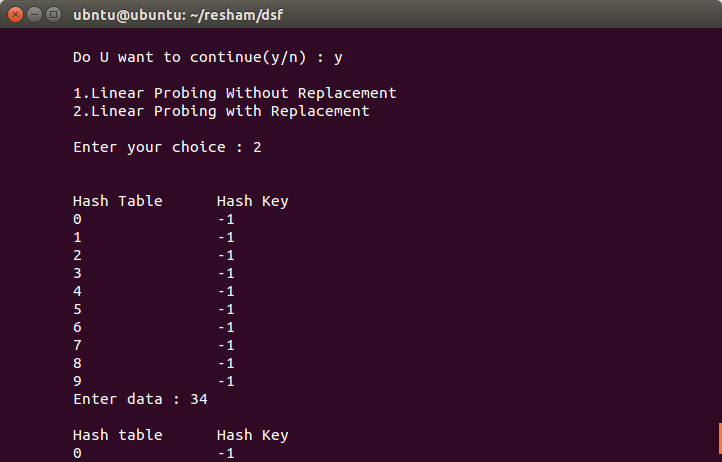
return 1;

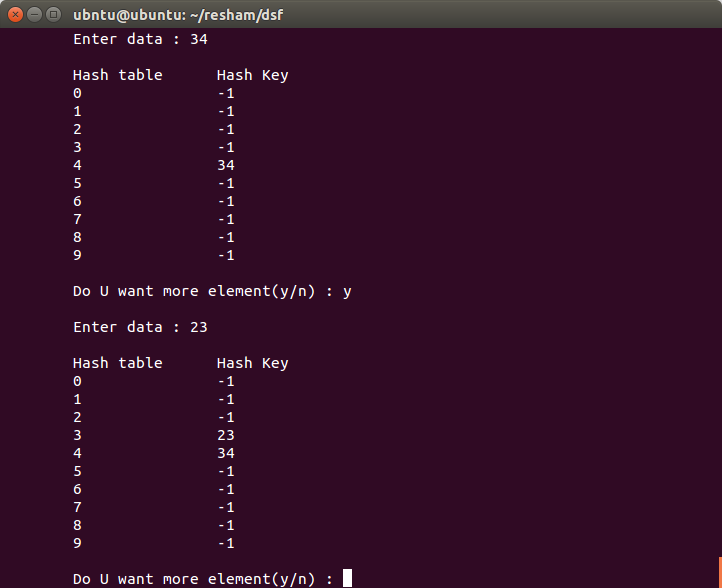
}

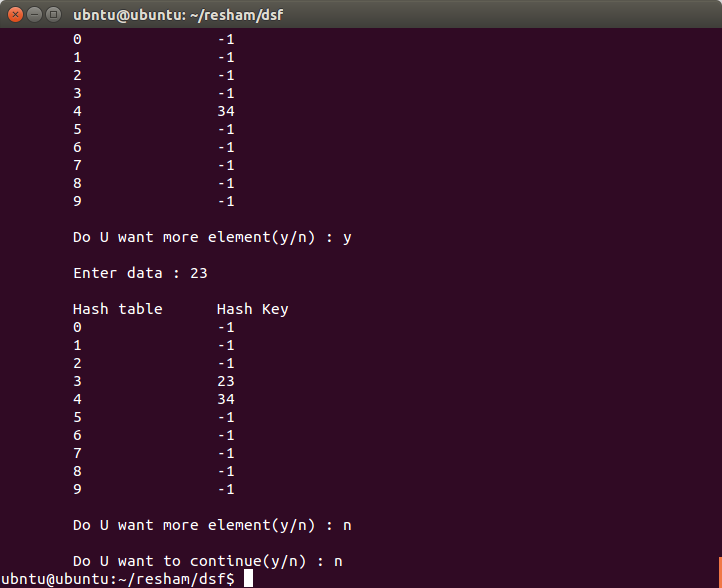
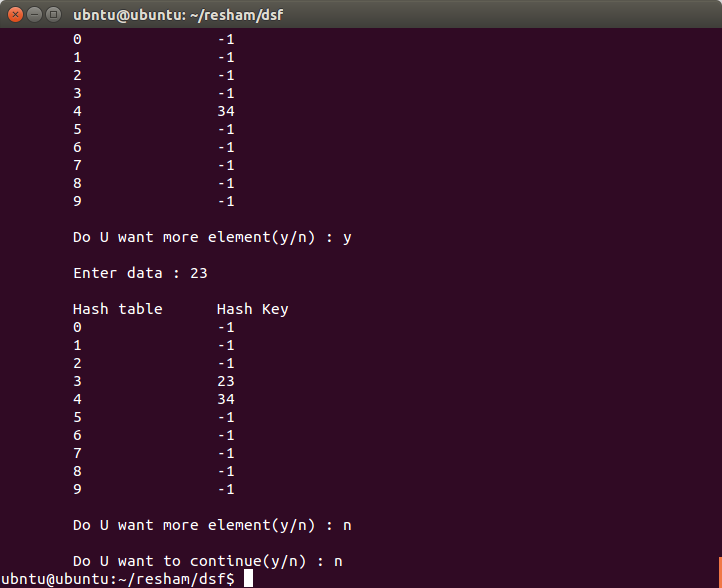
**Output :**

****

****

****

****

****