#include<iostream>

#define MAX 10

using namespace std;

class WO\_chain

{

private:

int a[MAX][2];

public:

WO\_chain();

int create(int);

void chain(int,int),display();

};

WO\_chain::WO\_chain()

{

int i;

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

{

a[i][0]=-1;

a[i][1]=-1;

}

}

int WO\_chain::create(int key1)

{

int sum = 1,rem=0,n1;

while (key1 != 0)

{

rem = key1 % 10;

sum = sum \* rem;

key1 = key1 / 10;

}

n1=sum%10;

return n1;

}

void WO\_chain::chain(int key,int num)

{

int flag,i,count=0,ch;

flag=0;

//checking full condition

i=0;

while(i<MAX)

{

if(a[i][0]!=-1)

count++;

i++;

}

if(count==MAX)

{

cout<<"\nHash Table Is Full";

display();

exit(1);

}

//placing number otherwise

if(a[key][0]==-1)//no collision case

a[key][0]=num;

else //if collision occurs

{

ch=a[key][1];//taking the chain

//If only one number in hash table with current obtained key

if(ch==-1)

{

for(i=key+1;i<MAX;i++)//performing linear probing

{

if(a[i][0]==-1) //at immediate empty slot

{

a[i][0]=num;//placiing number

a[key][1]=i; //setting the chain

flag=1;

break;

}

}

}

//if many numbers are already in the hash table

//we will find the next empty slot to place number

else

{

while((a[ch][0]!=-1)&&(a[ch][1]!=-1))//traversing thro chain till empty slot is found\*/

ch=a[ch][1];

for(i=ch+1;i<MAX;i++)

{

if(a[i][0]==-1)

{

a[i][0]=num;//placing the number

a[ch][1]=i; //setting chain

flag=1;

break;

}

}

}

//If the numbers are occupied somewhere from middle and are stored upto

//the MAX then we will search for the empty slot upper half of the array

if(flag!=1)

{

if(ch==-1)

{

for(i=0;i<key;i++)//performing linear probing

{

if(a[i][0]==-1) //at immediate empty slot

{

a[i][0]=num;//placiing number

a[key][1]=i; //setting the chain

flag=1;

break;

}

}

}

//if many numbers are already in the hash table

//we will find the next empty slot to place number

else

{

//traversing thro chain till empty slot is found

while((a[ch][0]!=-1)&&(a[ch][1]!=-1))

ch=a[ch][1];

for(i=ch+1;i<key;i++)

{

if(a[i][0]==-1)

{

a[i][0]=num;//placing the number

a[ch][1]=i; //setting chain

flag=1;

break;

}

}

}

}

}

}

void WO\_chain::display()

{

int i;

cout<<"\n The Hash Table is...\n";

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

cout<<"\n "<<i<<" "<<a[i][0]<<" "<<a[i][1];

}

int main()

{

int num,key,i;

char ans;

WO\_chain h;

//clrscr();

cout<<"\nChaining Without Replacement";

do

{

cout<<"\n Enter The Number";

cin>>num;

key=h.create(num);//returns hash key

h.chain(key,num);//collision handled by chaining without replacement

cout<<"\n Do U Wish To Continue?(y/n)";

cin>>ans;

}while(ans=='y');

h.display();//displays hash table

}

/\*output

Chaining Without Replacement

Enter The Number23

Do U Wish To Continue?(y/n)y

Enter The Number05

Do U Wish To Continue?(y/n)n

The Hash Table is...

0 -1 -1

1 -1 -1

2 -1 -1

3 -1 -1

4 -1 -1

5 5 -1

6 23 -1

7 -1 -1

8 -1 -1

9 -1 -1\*/