Practical No. 02

1. Searching Techniques :-
   1. Linear Search
   2. Binary Search

Code:-

#include<iostream>

#define max 99

using namespace std;

int n;

int A[max];

class LinearSearch

{

int temp,count;

public:

LinearSearch()

{

count = 0;

temp = 0;

}

void getdata()

{

cout<<"Enter the size of the Array"<<endl;

cin>>n;

cout<<"Enter the values of the Array"<<endl;

for(int i=0; i<n; i++)

{

cin>>A[i];

}

}

void display()

{

cout<<endl;

for(int i=0; i<n; i++)

{

cout<<A[i]<<"\t";

}

cout<<endl;

}

void search(int x)

{

for(int i=0; i<n; i++)

{

if(x == A[i])

{

count++;

cout<<"\nElement has been found at position = "<<i+1<<endl;

display();

break;

}

}

if(count == 0)

{

cout<<"\nElement Not Found"<<endl;

A[n]=x;

cout<<endl;

for(int i=0; i<=n; i++)

{

cout<<A[i]<<"\t";

}

cout<<endl;

n++;

}

}

void sort()

{

for(int i=0; i<n-1; i++)

{

count = 0;

for(int j=0; j<n-i-1; j++)

{

if(A[j]>A[j+1])

{

temp = A[j];

A[j] = A[j+1];

A[j+1] = temp;

count++;

}

}

if(count == 0)

{

break;

}

}

display();

}

int bsearch(int x)

{

int low = 0;

int high = n;

while(low<=high)

{

int mid=(low+high)/2;

if(A[mid]<x)

{

low=mid+1;

}

else if(A[mid]>x)

{

high=mid-1;

}

else

{

return mid;

}

}

return -1;

}

};

int main()

{

LinearSearch ls;

int s,c=0,k=-1;

ls.getdata();

ls.display();

for( ; ; ){

cout<<"Options :\n";

cout<<"\nEnter 1 to Sort.\nEnter 2 for a Linear Search.\nEnter 3 for a Binary Search.\nEnter 4 to Display Array.\nEnter 9 to Exit.\n";

cin>>c;

if(c==9)

{

cout<<"Bye!!!"<<endl;

break;

}

switch(c)

{

case 2:

cout<<"Enter Any Integer Number To Search If It's There You'll Get Results Otherwise It'll be Added To The Array\n";

cin>>s;

ls.search(s);

break;

case 1:

ls.sort();

break;

case 3:

cout<<"Enter Any Integer Number To Search\n"<<endl;

cin>>s;

ls.sort();

k = ls.bsearch(s);

if(k!=-1)

{

cout<<"Element found at position "<<k+1<<endl;

}

else

{

cout<<"Element not found"<<endl;

}

break;

case 4:

ls.display();

break;

default:

cout<<"Invalid Number"<<endl;

break;

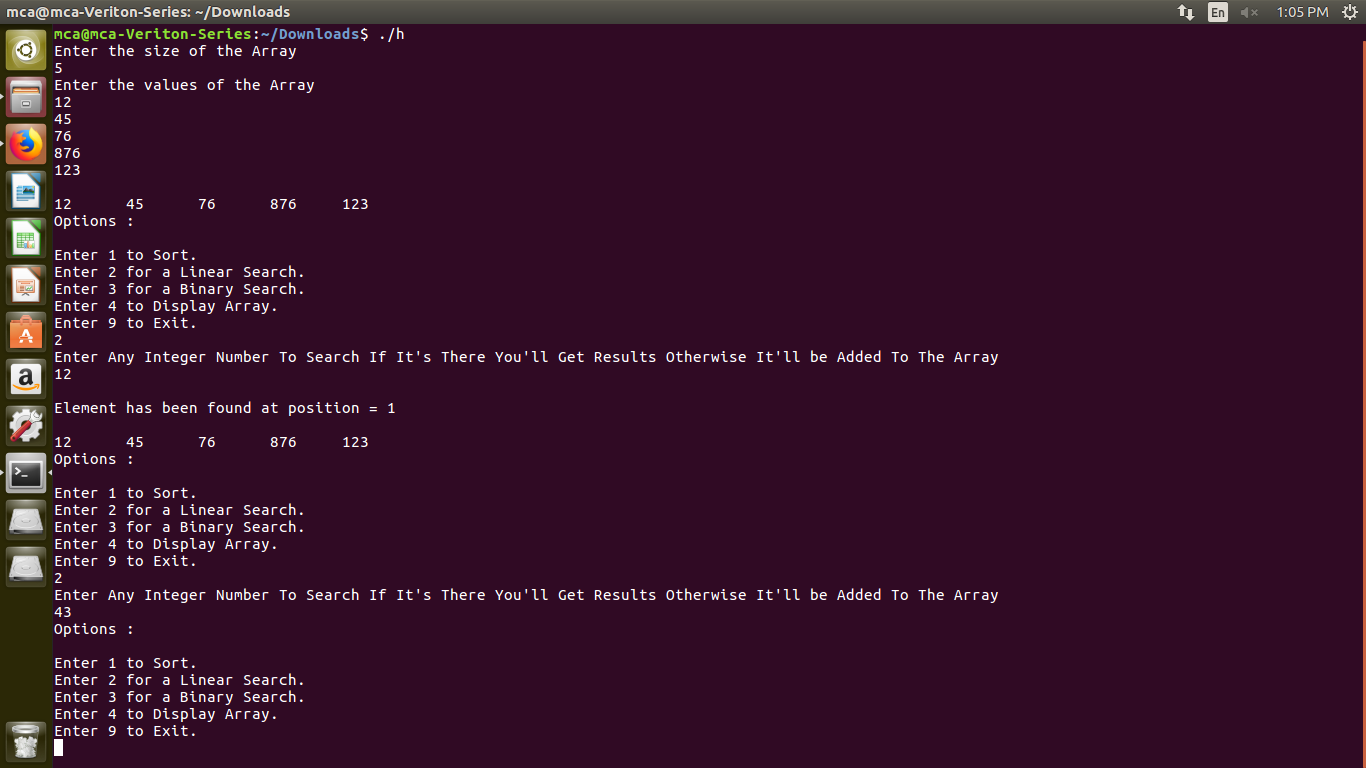
}

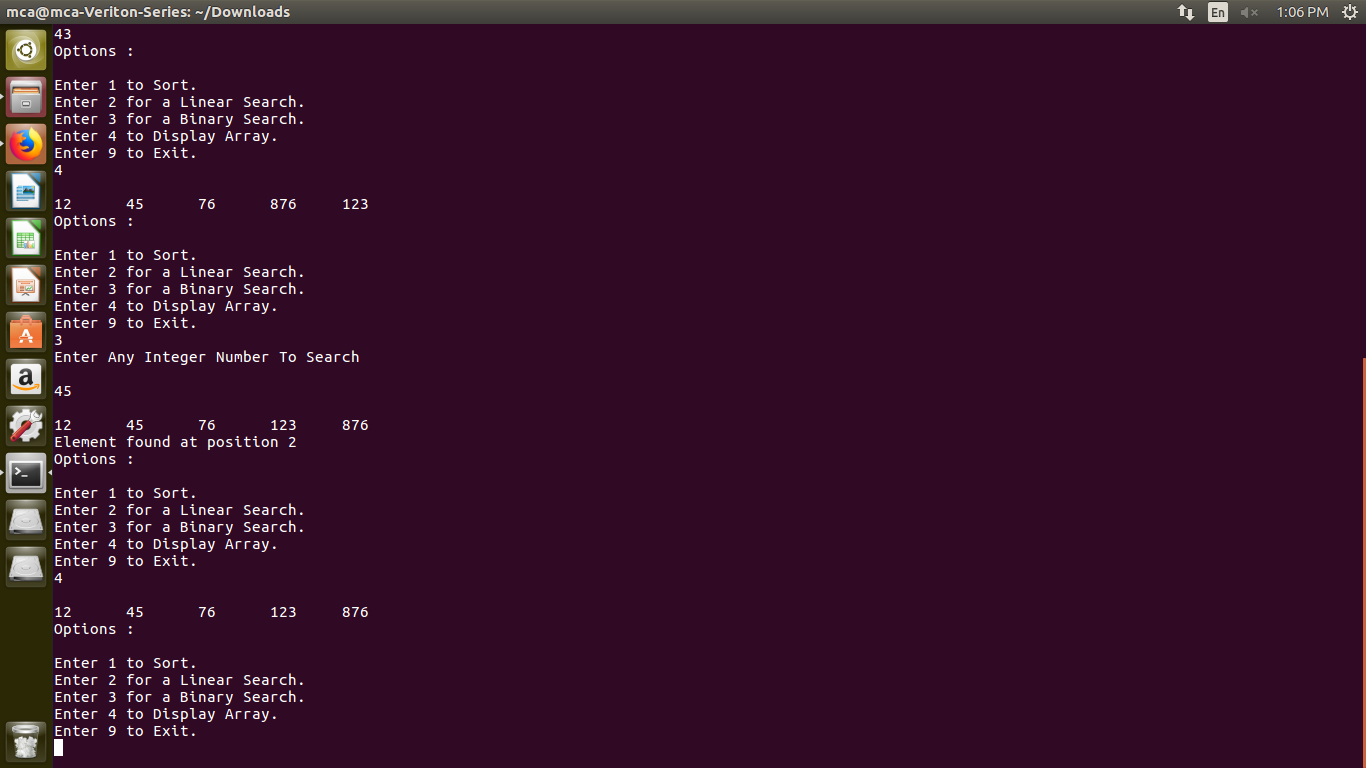
}

return 0;

}

Snapshot:-



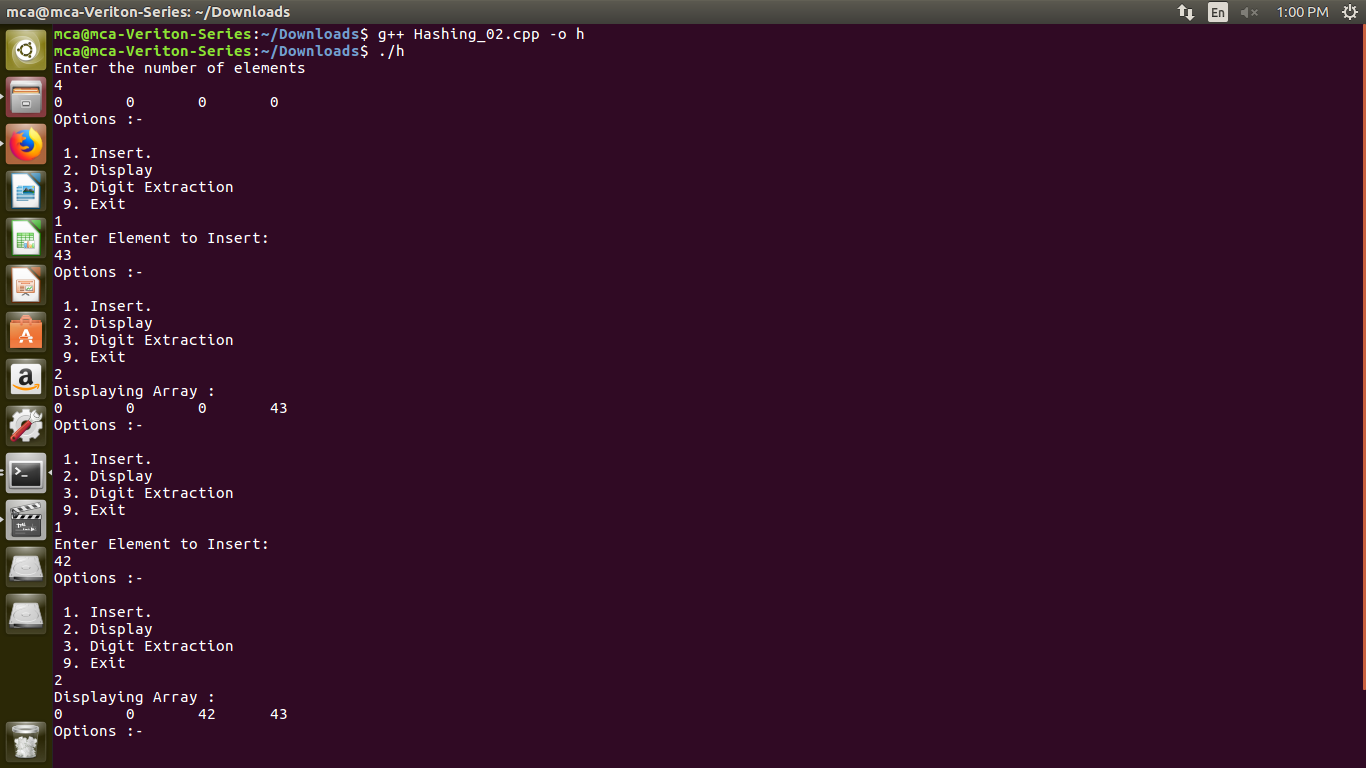


1. Hashing :-
   1. Modulo Division

Code :-

#include<iostream>  
#define max 25  
using namespace std;  
  
class Hashing  
{  
 public:  
 int A[max],n,count;  
 Hashing()  
 {  
 n = 0;  
 count = 0;  
 }  
  
 void getdata()  
 {  
 cout<<"Enter the number of elements"<<endl;  
 cin>>n;  
 for(int i=0; i<n; i++)  
 {  
 A[i]=0;  
 }  
 }  
  
 void display()  
 {  
 for(int i=0; i<n; i++)  
 {  
 cout<<A[i]<<"\t";  
 }  
 cout<<endl;  
 }  
  
 void insert(int key)  
 {  
 int offset, rem, res;  
 res = key % n;  
 int temp = isEmpty(res);  
 if(temp != 100)  
 {  
 A[temp] = key;  
 }  
 else  
 {  
 cout<<"Array is full"<<endl;  
 }  
 }  
   
 int isEmpty(int res)  
 {  
   
 for (int i = res; i<n; i++)   
 {  
 if(A[i] == 0)  
 {  
 return i;  
 break;  
 }  
 else  
 {  
 count = n;  
 }  
 }  
 if(count == n)  
 {  
 for(int i = 0; i<res; i++)  
 {  
 if(A[i] == 0)  
 {  
 return i;  
 break;  
 }  
 }  
 }  
 return 100;  
 }  
  
};  
  
int main()  
{  
int x,i;  
Hashing h;  
h.getdata();  
h.display();  
for(;;)  
{  
 cout<<"Options :- "<<endl;  
 cout<<"\n 1. Insert.\n 2. Display\n 3. Digit Extraction \n 9. Exit"<<endl;  
 cin>>x;  
 if(x == 9)  
 {  
 cout<<"Bye!!!"<<endl;  
 break;  
 }  
 switch(x)  
 {  
 case 1:  
 cout<<"Enter Element to Insert:"<<endl;  
 cin>>i;  
 h.insert(i);  
 break;  
 case 2:  
 cout<<"Displaying Array : "<<endl;  
 h.display();  
 break;  
 default:  
 cout<<"Invalid Options"<<endl;  
 break;  
 }  
}  
return 0;  
}

Screenshot :-

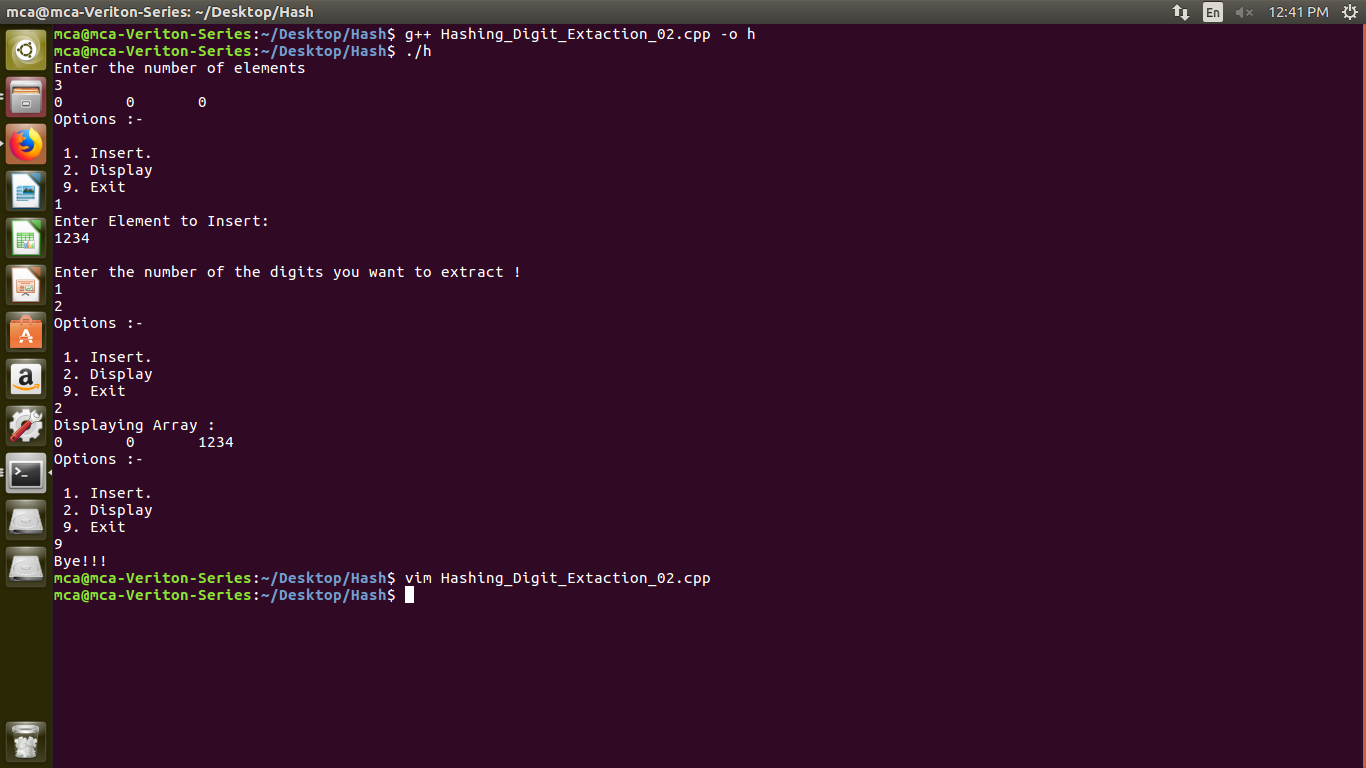


* 1. Digit Extraction

Code :-

#include<iostream>  
#define max 100  
using namespace std;  
  
int ncount = 0;  
class Hashings  
{  
 public:  
 long int A[max], n, count;  
 Hashings()  
 {  
 n = 0;  
 count = 0;  
 }  
  
 void getdata()  
 {  
 cout<<"Enter the number of elements"<<endl;  
 cin>>n;  
 for(int i=0; i<n; i++)  
 {  
 A[i]=0;  
 }  
 }  
  
 void display()  
 {  
 for(int i=0; i<n; i++)  
 {  
 cout<<A[i]<<"\t";  
 }  
 cout<<endl;  
 }  
  
 void insert(int key)  
 {  
 //Digit counter for Key  
 int res = 0, kcount = 0, te[max], digit = 0, s = 0;  
 cout<<"\nEnter the number of the digits you want to extract !\n";  
 cin>>digit;  
  
 int ndigit[digit];  
   
 for(int i = 0; i<digit; i++)  
 {  
 cin>>ndigit[i];  
 }  
   
 int l = key;  
  
 for(int i=0; l>0; i++)  
 {  
 te[i] = l % 10;  
 l = l / 10;  
 s++;   
 }  
 int kel[s];  
  
 for(int j=s-1, k = 0; j>=0; j--, k++)  
 {  
 kel[k] = te[j];  
 }  
   
 int c=1;  
 for(int i = 0; i<digit; i++)  
 {  
 res = res + kel[ndigit[i]-1]\*c;  
 c=c\*10;  
 }  
   
 res = res % n;  
  
 int temp = isEmpty(res);  
 if(temp != 100)  
 {  
 A[temp] = key;  
 }  
 else  
 {  
 cout<<"Array is full"<<endl;  
 }  
 }  
   
 int isEmpty(int res)  
 {  
   
 for (int i = res; i<n; i++)   
 {  
 if(A[i] == 0)  
 {  
 return i;  
 break;  
 }  
 else  
 {  
 count = n;  
 }  
 }  
 if(count == n)  
 {  
 for(int i = 0; i<res; i++)  
 {  
 if(A[i] == 0)  
 {  
 return i;  
 break;  
 }  
 }  
 }  
 return 100;  
 }  
  
};  
  
int main()  
{  
  
 int x,i;  
 Hashings h;  
 h.getdata();  
 h.display();  
 for(;;)  
 {  
 cout<<"Options :- "<<endl;  
 cout<<"\n 1. Insert.\n 2. Display \n 9. Exit"<<endl;  
 cin>>x;  
 if(x == 9)  
 {  
 cout<<"Bye!!!"<<endl;  
 break;  
 }  
 switch(x)  
 {  
 case 1:  
 cout<<"Enter Element to Insert:"<<endl;  
 cin>>i;  
 h.insert(i);  
 break;  
 case 2:  
 cout<<"Displaying Array : "<<endl;  
 h.display();  
 break;  
 default:  
 cout<<"Invalid Options"<<endl;  
 break;  
 }  
 }  
return 0;  
}

Screenshot :-

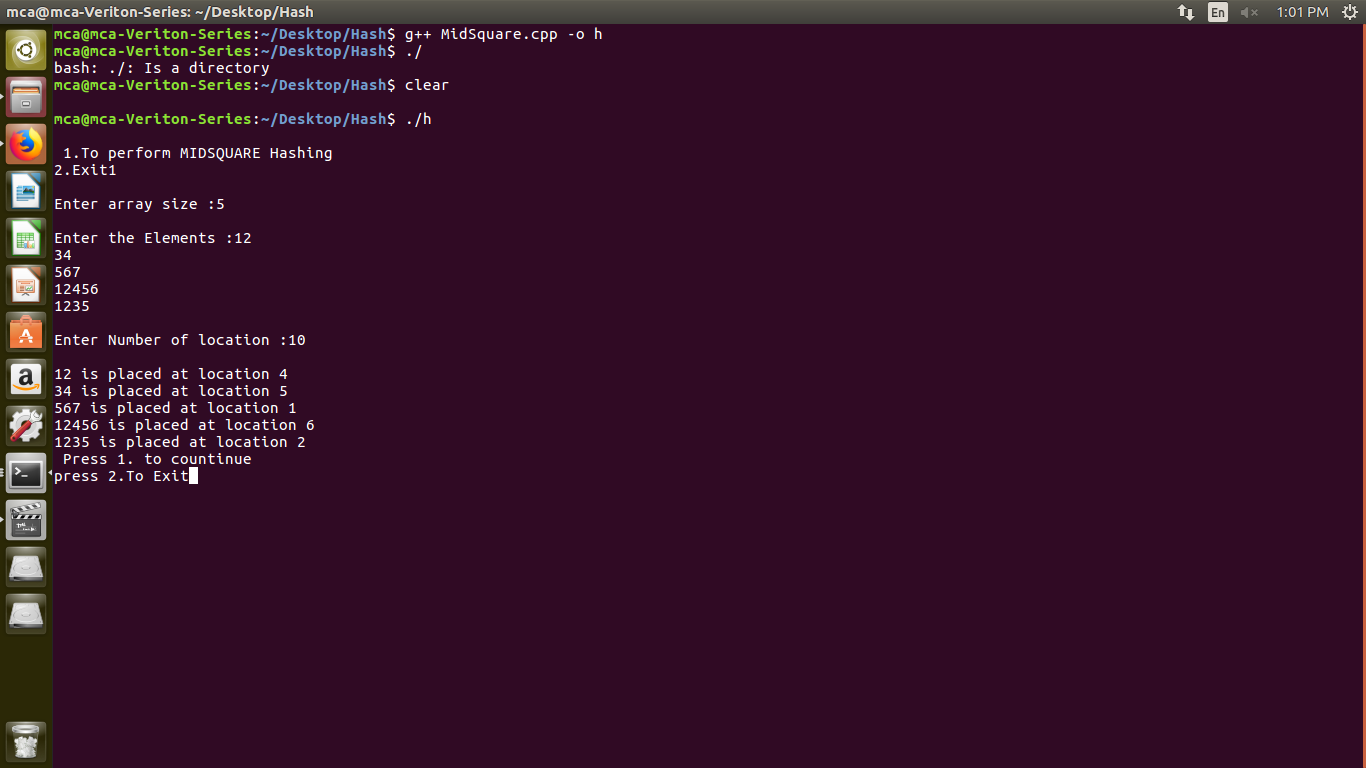


* 1. MidSquare

Code :-

#include<iostream>  
#include<stdlib.h>  
#define max 100  
using namespace std;  
class hashtechniques  
{  
 public:  
 int n,nl,a1[max],a2[max],i,j,k,ch,res,co,ea,dig,div,pos;  
 void getdata()  
 {  
 cout<<endl<<"Enter array size :";  
  
 cin>>n;  
 cout<<endl<<"Enter the Elements :";  
 for(i=0;i<n;i++)  
 cin>>a1[i];  
 cout<<endl<<"Enter Number of location :";  
 cin>>nl;  
 for(i=0;i<nl;i++)  
 a2[i]=0;  
 //menu();  
 //MidSquare();  
 }  
 void menu()  
 {  
 cout<<"\n 1.To perform MIDSQUARE Hashing"<<endl<<"2.Exit";  
 cin>>ch;  
 switch(ch)  
 {  
 case 1: getdata();  
 midsqr();  
 break;  
 case 2:cout<<"Bye !!!";  
 break;  
  
 default:  
 cout<<"\n Enter Proper choice:";  
 menu();  
 }  
 }  
  
 void select()  
 {  
 cout<<"\n Press 1. to countinue "<<endl<<"press 2.To Exit";  
 cin>>res;  
 if(res==1)  
 {  
 menu();  
 }  
 else  
 {  
 exit(0);  
 }  
 }  
  
 void midsqr()  
 {  
 for(i=0;i<n;i++)  
 {  
 k=a1[i]\*a1[i];  
 int num=k;  
 int flag=0;  
 int div=1;  
 int dig=0;  
 while(num>0)  
 {  
 num=num/10;  
 dig++;  
 div=div\*10;  
 }  
 div=div/100;  
 while(flag==0)  
 {  
 while(k>=0&&k<nl)  
 {  
 if(a2[k]==0)  
 {  
 a2[k]=a1[i];  
 cout<<endl<<a1[i]<<" is placed at location "<<k;  
 flag=1;  
 break;  
 }  
 else  
 {  
 k=k+1;  
 }  
 }  
 if(flag==0)  
 {  
 if(k>99)  
 {  
 k=k/10;  
 k=k%div;  
 div=div/10;  
 }  
 else  
 {  
 k=k%10;  
 }  
 }  
  
 }  
  
 }  
 select();  
 }  
};  
  
  
int main()  
{  
 hashtechniques h;  
 h.menu();  
 return 0;  
}

Screenshot :-



* 1. Fold Boundary

Code :-

#include<iostream>  
#include<cmath>  
#define max 100  
  
using namespace std;  
  
class hashing  
{  
  
 public:  
 int a[max],b[max];  
 int i,j,k,n,n1,c,m,len,c1,rem,d,s,sum,num,num1,l\_num1,rev,digit,p,ans,num3,num2,num4,ch;  
 int t;  
  
 void menu()  
 {  
 ch=0;  
 while(ch!=3)  
 {  
  
 cout<<endl<<endl;  
  
 cout<<"1.Display"<<endl;  
  
 cout<<"2.Fold Boundary"<<endl;  
  
 cout<<"3.Exit"<<endl;  
  
 cout<<"Enter options "<<endl;  
  
 cin>>ch;  
 switch(ch)  
  
 {  
 case 1:display();  
 break;  
 case 2:foldb();  
 break;  
 default:  
 cout<<"invalid choice"<<endl;  
  
 }  
 }  
  
 }  
  
 void getdata()  
 {  
 cout<<"Enter the size of array "<<endl;  
 cin>>n;  
 cout<<"Enter the element of array "<<endl;  
 for(i=0;i<n;i++)  
 {  
 cin>>a[i];  
  
 }  
 cout<<"Enter size of location "<<endl;  
  
 cin>>n1;  
 for(i=0;i<n1;i++)  
 {  
 b[i]=0;  
 }  
// display();  
 menu();  
//s foldf();  
 }  
  
 void display()  
 {  
 cout<<"Elements are :"<<endl;  
 for(i=0;i<n;i++)  
 {  
 cout<<a[i]<<" ";  
 }  
//foldf();  
 }  
  
 int length(int l)  
 {  
 j=0;  
 while(c>0)  
 {  
 c=c/10;  
 j++;  
 }  
 return j;  
 }  
  
 int reverse(int r)  
 {  
 int rem,rev=0;  
 while(r>0)  
 {  
 rem=r%10;  
 r=r/10;  
 rev=(rev\*10)+rem;  
 }  
  
 return rev;  
 }  
  
 int foldb()  
 {  
 for(i=0;i<n;i++)  
 {  
 k=a[i];  
 num=k;  
 int p=0;  
 while(k>0)  
 {  
 k=k/10;  
 p++;  
 }  
  
 c=n1-1;  
  
 c1=length(c);  
  
 t=(p/ c1) + ((p % c1)!= 0);  
  
 d=1;  
  
 sum=0;  
 while(num>0)  
 {  
  
 s=pow(10,c1);  
 num1=num %s;  
  
 num2=num1;  
  
 m=0;  
  
 while(num1>0)  
  
 {  
  
 num1=num1/10;  
  
 m++;  
  
 }  
  
 if(m==1)  
  
 {  
  
 sum=sum+(num2\*10);  
  
 break;  
  
 }  
  
 if(d==1 || d==t)  
  
 {  
  
 num3=reverse(num2);  
  
 sum=sum+num3;  
  
 }  
 else  
 {  
  
 sum=sum+num2;  
  
 }  
  
 num=num/pow(10,c1);  
  
 d=d+1;  
  
 }  
  
 if(sum>n1)  
  
 {  
  
 sum=sum%n1;  
  
 }  
  
 while(sum<n1)  
  
 {  
 if(b[sum]==0)  
  
 {  
 b[sum]=a[i];  
 cout<<endl<<a[i]<<"is placed at location "<<sum<<endl;  
 break;  
 }  
 else  
 {  
 sum++;  
 }  
 if(sum==n1)  
 sum=0;  
  
 }  
  
 }  
  
 }  
};  
  
int main()  
{  
  
hashing h;  
h.getdata();  
  
return 0;  
  
}

Screenshot :-

