**1) Write a menu driven program to demonstrate Linear search.**

**Code :**

**#include<iostream>**

**using namespace std;**

**class LSearch**

**{**

**private:**

**int noe,key,i;**

**int arr[100];**

**public:**

**void GetData()**

**{**

**cout<<"Enter The Size of List(Maximum 100) : ";**

**cin>>noe;**

**for(i=0;i<noe;i++)**

**{**

**cout<<"Enter Data : ";**

**cin>>arr[i];**

**}**

**cout<<endl<<"Enter Number to be Searched : ";**

**cin>>key;**

**}**

**void add()**

**{**

**noe+=1;**

**arr[noe-1]=key;**

**}**

**void ShowData()**

**{**

**for(int i=0;i<noe;i++)**

**{**

**cout<<arr[i]<<" ";**

**}**

**}**

**int Search()**

**{**

**int flag=0;**

**for(i=0;i<noe;i++)**

**{**

**if(key==arr[i])**

**{**

**flag=1;**

**cout<<"The Number is Found At : "<<i<<"th Index and "<<i+1<<"th position"<<endl;**

**break;**

**}**

**}**

**return flag;**

**}**

**};**

**int main()**

**{**

**LSearch b=LSearch();**

**b.GetData();**

**if(b.Search() == 0)**

**{**

**b.add();**

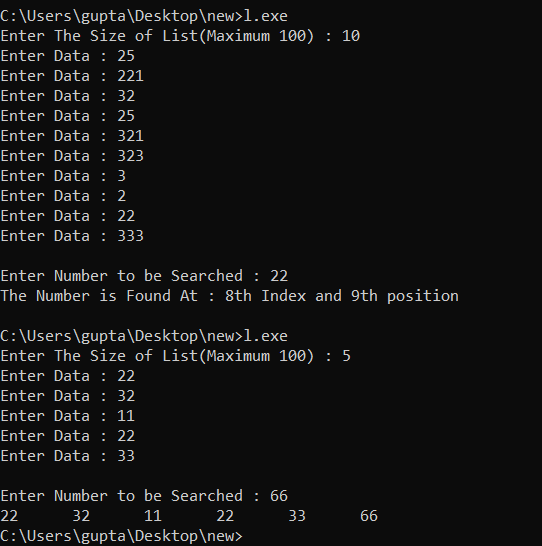
**b.ShowData();**

**}**

**return 0;**

**}**

**Output :**

****

**2) Write a menu driven program to demonstrate Binary search.**

**Code :**

**#include<iostream>**

**using namespace std;**

**class BSearch**

**{**

**private:**

**int noe,key,low,high,mid,i,j,temp;**

**int arr[100];**

**public:**

**void GetData()**

**{**

**cout<<"Enter The Size of List(Maximum 100) : ";**

**cin>>noe;**

**for(i=0;i<noe;i++)**

**{**

**cout<<"Enter Data : ";**

**cin>>arr[i];**

**}**

**}**

**int Sorted()**

**{**

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

**{**

**if(arr[i]>arr[i+1])**

**return 0;**

**}**

**return 1;**

**}**

**void Sort()**

**{**

**for(i=0;i<noe;i++)**

**{**

**if(Sorted())**

**break;**

**for(j=0;j<noe-1;j++)**

**{**

**if(arr[j]>arr[j+1])**

**{**

**temp=arr[j];**

**arr[j]=arr[j+1];**

**arr[j+1]=temp;**

**}**

**}**

**}**

**}**

**void ShowData()**

**{**

**cout<<"Sorted Array :";**

**for(int i=0;i<noe;i++)**

**{**

**cout<<arr[i]<<" ";**

**}**

**cout<<"\n\n";**

**cout<<endl<<"Enter Number to be Searched : ";**

**cin>>key;**

**}**

**void Search()**

**{**

**int flag=0;**

**low=0;**

**high=noe-1;**

**while(low <= high)**

**{**

**mid=(low+high)/2;**

**cout<<mid;**

**if(arr[mid] == key)**

**{**

**flag=1;**

**cout<<"The Number is Found At : "<<mid<<"th Index and "<<mid+1<<"th position"<<endl;**

**break;**

**}**

**else if(arr[mid] > key)**

**{**

**high=mid-1;**

**}**

**else**

**{**

**low=mid+1;**

**}**

**}**

**if (flag == 0)**

**{**

**cout<<"Number Not Present";**

**}**

**}**

**};**

**int main()**

**{**

**BSearch b=BSearch();**

**b.GetData();**

**b.Sort();**

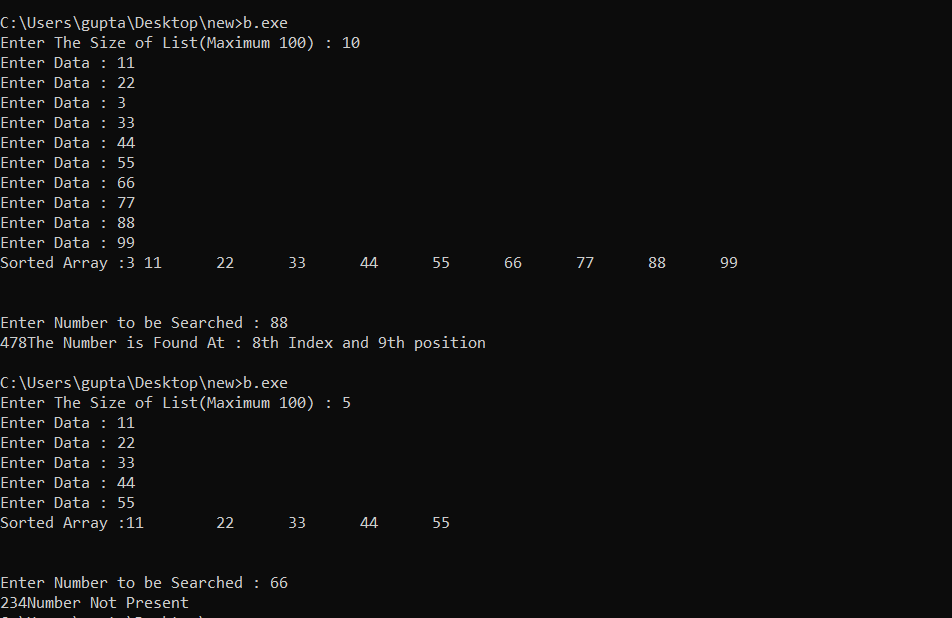
**b.ShowData();**

**b.Search();**

**return 0;**

**}**

**Output :**

****

**3) Write a menu driven program to demonstrate Modulo division with linear probing.**

**Code :**

**#include<iostream>**

**using namespace std;**

**class Modulo**

**{**

**public:**

**void Hash(int \*arr,int \*key\_arr,int n,int noe)**

**{**

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

**{**

**if(i>=noe)**

**{**

**cout<<"Out of Bounds"<<endl;**

**break;**

**}**

**int loc=key\_arr[i]%noe;**

**if(arr[loc] == 0)**

**{**

**arr[loc]=key\_arr[i];**

**}**

**else**

**{**

**while(1)**

**{**

**loc++;**

**if(loc >= noe)**

**{**

**loc=loc%noe;**

**}**

**if(arr[loc] == 0)**

**{**

**arr[loc]=key\_arr[i];**

**break;**

**}**

**}**

**}**

**}**

**}**

**void Display(int \*arr,int noe)**

**{**

**for(int i=0;i<noe;i++)**

**{**

**cout<<i<<" "<<arr[i];**

**cout<<endl;**

**}**

**}**

**};**

**int main()**

**{**

**int n,noe;**

**cout<<"Enter The Size of Array (Location Array) :";**

**cin>>noe;**

**int arr[noe] ={0};**

**cout<<"Enter The Number of Elements to be Hashed :";**

**cin>>n;**

**int key\_arr[n];**

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

**{**

**cout<<"Enter Data "<<i<<" : ";**

**cin>>key\_arr[i];**

**}**

**Modulo m=Modulo();**

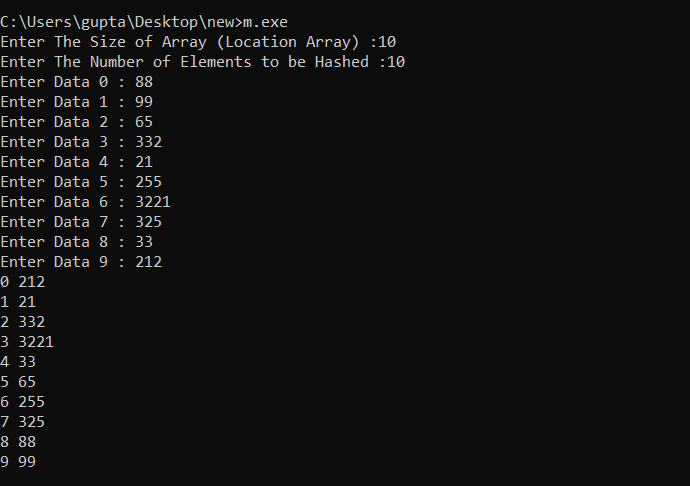
**m.Hash(arr,key\_arr,n,noe);**

**m.Display(arr,noe);**

**return 0;**

**}**

**Output :**

****

**4) Write a menu driven program to demonstrate Digit extraction with linear probing.**

**Code :**

**#include <iostream>**

**using namespace std;**

**class DigitExtraction**

**{**

**private:**

**int arr[100], extract\_arr[100], single\_arr[100], output\_array[100];**

**int choice, size, noe, n;**

**public:**

**void get()**

**{**

**cout << "Enter the Size of Location Array : "**

**<< " ";**

**cin >> size;**

**for (int i = 0; i < size; i++)**

**{**

**arr[i] = -1;**

**}**

**cout << "Enter the Number of locations you want to extract : "**

**<< " ";**

**cin >> n;**

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

**{**

**cout << "Enter Location : ";**

**cin >> extract\_arr[i];**

**}**

**cout << "How many numbers do you want to hash? : "**

**<< " ";**

**cin >> noe;**

**for (int i = 0; i < noe; i++)**

**{**

**cout << "Enter Element no " << i << " : ";**

**cin >> choice;**

**operations(choice);**

**}**

**}**

**void display()**

**{**

**for (int i = 0; i < size; i++)**

**{**

**if (arr[i] == -1)**

**{**

**}**

**else**

**{**

**cout << i<<" : "<<arr[i] << "\t";**

**}**

**}**

**cout << endl;**

**}**

**void operations(int choice)**

**{**

**int location;**

**int key = choice;**

**location = count(key);**

**cout << "Location is : "<<location << endl;**

**if (arr[location] == -1)**

**{**

**arr[location] = choice;**

**}**

**else**

**{**

**while (1)**

**{**

**location++;**

**if (arr[location] == -1)**

**{**

**arr[location] = choice;**

**break;**

**}**

**if (location >= size)**

**{**

**location = 0;**

**}**

**}**

**}**

**}**

**int count(int key)**

**{**

**int temp, loc, value = 0, num, count = 0;**

**temp = key;**

**while (temp > 0)**

**{**

**temp = temp / 10;**

**count++;**

**}**

**for (int i = 0; i < count; i++)**

**{**

**num = key % 10;**

**single\_arr[i] = num;**

**key = key / 10;**

**}**

**for(int k=0;k<n;k++)**

**{**

**int m=extract\_arr[k];**

**output\_array[k]=single\_arr[m-1];**

**}**

**int start=0;**

**int end=n-1;**

**while (start < end)**

**{**

**int temp = output\_array[start];**

**output\_array[start] = output\_array[end];**

**output\_array[end] = temp;**

**start++;**

**end--;**

**}**

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

**{**

**value = value \* 10 + output\_array[i];**

**}**

**loc = value % size;**

**return loc;**

**}**

**};**

**int main()**

**{**

**DigitExtraction d;**

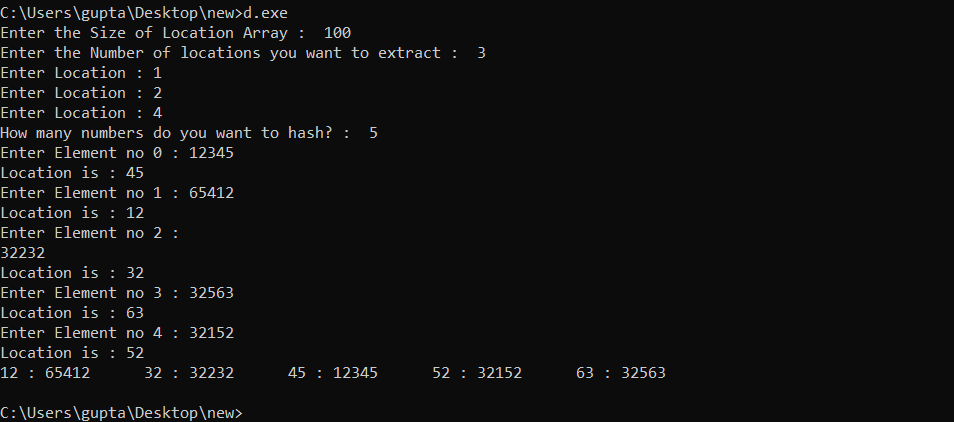
**d.get();**

**d.display();**

**return 0;**

**}**

**Output :**

****

**5) Write a menu driven program to demonstrate Fold boundary with linear probing .**

**//Manish,Shivam and I have created this program together**

**Code :**

**#include <iostream>**

**#include <math.h>**

**#include<cmath>**

**using namespace std;**

**class Hash**

**{**

**int arry[1000];**

**int final[1000];**

**int no\_ele, size,divs, div\_size;**

**public:**

**Hash(int n, int s)**

**{**

**no\_ele = n;**

**size = s;**

**for (int i = 0; i < s; i++)**

**final[i] = -1;**

**}**

**void getdata()**

**{**

**cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";**

**for (int i = 0; i < no\_ele; i++)**

**{**

**cout << "Enter element : ";**

**cin >> arry[i];**

**if(arry[i]<0)**

**{**

**cout<<"\n----- Numbers cannot be negative -----\n";**

**cout<<"Enter a positive number : ";**

**cin>>arry[i];**

**}**

**}**

**}**

**int linear\_probing(int index)**

**{**

**while (final[index] != -1)**

**{**

**index++;**

**if (index == size)**

**index = 0;**

**}**

**return index;**

**}**

**//HASHING FUNCTION**

**void fold\_boundry()**

**{**

**cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";**

**int index;**

**for (int i = 0; i < no\_ele; i++)**

**{**

**cout<<"\n--- Element "<<i+1<<" ---";**

**if (i > size - 1)**

**{**

**cout << "\n--- " << arry[i] << " cannot be stored as all locations are full ---\n";**

**}**

**else**

**{**

**index = divide(arry[i]);**

**cout << "\nLocation = " << index << "\n";**

**if (final[index] == -1)**

**final[index] = arry[i];**

**else**

**{**

**modulo\_division(index, i);**

**}**

**}**

**}**

**}**

**void modulo\_division(int index, int i)**

**{**

**index = index % size;**

**if (final[index] != -1)**

**index = linear\_probing(index);**

**final[index] = arry[i];**

**cout << "Location after Modulo-Division = " << index << "\n";**

**cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";**

**}**

**//Functions for fold boundry**

**int cnt(int n)**

**{**

**int temp = n, count = 0;**

**while (temp > 0)**

**{**

**count++;**

**temp /= 10;**

**}**

**return count;**

**}**

**int large(int a[])**

**{**

**int big = a[0];**

**for (int i = 1; i < divs; i++)**

**{**

**if (big < a[i])**

**big = a[i];**

**}**

**return big;**

**}**

**int rev(int n)**

**{**

**int r = 0;**

**while (n > 0)**

**{**

**r = (r \* 10) + (n % 10);**

**n /= 10;**

**}**

**}**

**int rev\_arr(int n)**

**{**

**int r = 0;**

**while (n > 0)**

**{**

**r = (r \* 10) + (n % 10);**

**n /= 10;**

**}**

**if (cnt(r) == 1)**

**return r \* 10;**

**else**

**return r;**

**}**

**int divide(int n)**

**{**

**int a\_size = cnt(n), temp = n;**

**div\_size = cnt(size-1);**

**int temp\_size = a\_size;**

**int arr[a\_size];**

**int r;**

**while (temp\_size > 0)**

**{**

**arr[temp\_size - 1] = temp % 10;**

**temp /= 10;**

**temp\_size--;**

**}**

**cout << "\nDivided element : " << endl;**

**for (int i = 0; i < a\_size; i++)**

**cout << arr[i] << "\t";**

**divs =ceil((float) cnt(n) / cnt(size - 1));**

**int diffrent[divs];**

**temp\_size = divs - 1;**

**int count = 0;**

**temp = 0;**

**for (int i = a\_size - 1; i >= 0; i--)**

**{**

**temp = arr[i] + temp \* 10;**

**diffrent[temp\_size] = temp;**

**count++;**

**if (count == div\_size)**

**{**

**temp\_size--;**

**count = 0;**

**temp = 0;**

**}**

**}**

**if(div\_size>1)**

**{**

**for (int i = 0; i < divs; i++)**

**{**

**if (i != 0)**

**{**

**if (i != divs - 1)**

**diffrent[i] = rev\_arr(diffrent[i]);**

**}**

**}**

**}**

**if(diffrent[0]<10)**

**diffrent[0]\*=10;**

**//cout<<"\n\n+++++++ Last element : "<<diffrent[divs]**

**if(diffrent[divs-1]<10)**

**diffrent[divs-1]\*=10;**

**cout << "\n\nDiffrentiated elements : \n";**

**for (int i = 0; i < divs; i++)**

**cout << diffrent[i] << " ";**

**cout << "\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*";**

**return without\_carry(diffrent);**

**}**

**int without\_carry(int arr[])**

**{**

**int cn = cnt(large(arr)), add = 0, divisor = 1;**

**for (int i = 0; i < divs; i++)**

**{**

**add = add + arr[i];**

**}**

**while (cn > 0)**

**{**

**divisor = divisor \* 10;**

**cn--;**

**}**

**cout<<"\nAdd with carry = "<<add;**

**cout<<"\nAdd withiut carry ="<<add%divisor;**

**return add % divisor;**

**}**

**void showdata()**

**{**

**cout << "\nStored elements : \n";**

**for (int i = 0; i < size; i++)**

**{**

**if (final[i] != -1)**

**cout << "| " << i << " | " << final[i] << " |\n";**

**}**

**cout << endl;**

**}**

**};**

**int main()**

**{**

**int no\_ele, size;**

**cout << "\n\nEnter No. of loactions : ";**

**cin >> size;**

**cout << "Enter no of elements : ";**

**cin >> no\_ele;**

**Hash h(no\_ele, size);**

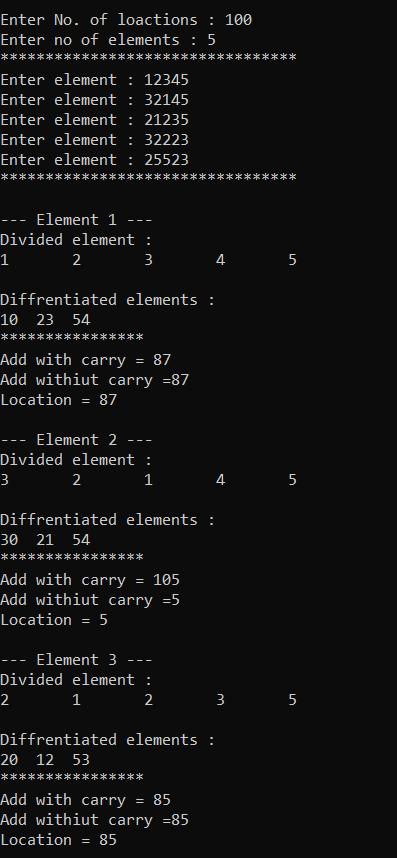
**h.getdata();**

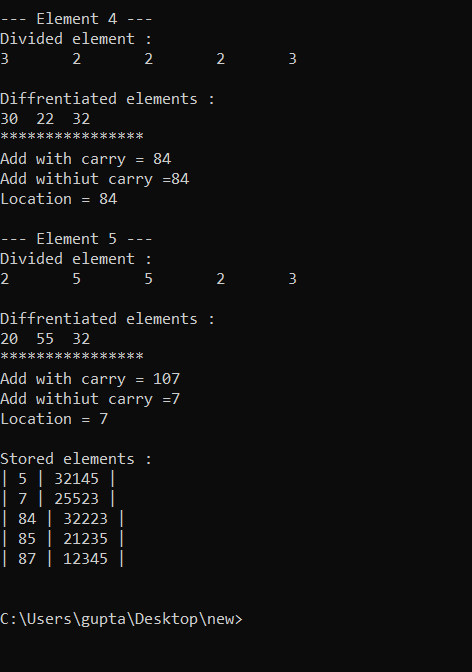
**h.fold\_boundry();**

**h.showdata();**

**}**

**Output :**

****

****

**6) Write a menu driven program to demonstrate Mid square with linear probing.**

**Code :**

**#include<iostream>**

**using namespace std;**

**class Mid**

**{**

**private:**

**int arr[100];**

**int choice,size,noe;**

**public:**

**void get()**

**{**

**cout<<"Enter the Size of Location Array : "<<" ";**

**cin>>size;**

**for(int i=0;i<size;i++)**

**{**

**arr[i]=-1;**

**}**

**cout<<"How many numbers do you want to hash? : "<<" ";**

**cin>>noe;**

**for(int i=0;i<noe;i++)**

**{**

**cout<<"Enter Element no "<<i<<" : ";**

**cin>>choice;**

**operations(choice);**

**}**

**}**

**void display()**

**{**

**for (int i = 0; i < size; i++)**

**{**

**if (arr[i] == -1)**

**{**

**}**

**else**

**{**

**cout << i<<" : "<<arr[i] << "\t";**

**}**

**}**

**cout << endl;**

**}**

**void operations(int choice)**

**{**

**int key,location;**

**key=choice\*choice;**

**location=count(key);**

**cout<<location<<endl;**

**if(arr[location] == -1)**

**{**

**arr[location]=choice;**

**}**

**else**

**{**

**while(1)**

**{**

**location++;**

**if(arr[location]==-1)**

**{**

**arr[location]=choice;**

**break;**

**}**

**if(location>=size)**

**{**

**location=0;**

**}**

**}**

**}**

**}**

**int count(int key)**

**{**

**int loc,temp,count=0;**

**temp=key;**

**while(temp>0)**

**{**

**temp=temp/10;**

**count++;**

**}**

**if(count%2==0)**

**{**

**for(int i=1;i<((count+1)/2);i++)**

**{**

**key=key/10;**

**}**

**loc=key%100;**

**loc=loc%size;**

**}**

**else if(count%2!=0)**

**{**

**for(int i=1;i<((count+1)/2);i++)**

**{**

**key=key/10;**

**}**

**loc=key%10;**

**}**

**return loc;**

**}**

**};**

**int main()**

**{**

**Mid m;**

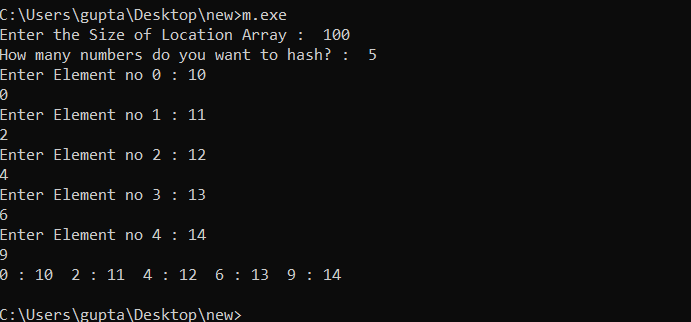
**m.get();**

**m.display();**

**return 0;**

**}**

**Output :**

****