**1.1 Bubble\_Sort\_17.cpp**

**#include<iostream>**

**using namespace std;**

**class BubbleSort**

**{**

**private:**

**int noe,i,j,temp;**

**int arr[100];**

**public:**

**void GetData()**

**{**

**cout<<"Enter The Number Of Elements Required To be Sorted : ";**

**cin>>noe;**

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

**{**

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

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

**}**

**}**

**void ShowData()**

**{**

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

**{**

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

**}**

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

**}**

**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;**

**}**

**}**

**cout<<"Pass "<<i<<": ";**

**ShowData();**

**}**

**}**

**};**

**int main()**

**{**

**BubbleSort b=BubbleSort();**

**b.GetData();**

**cout<<endl<<"Before Sorting :"<<endl;**

**b.ShowData();**

**cout<<endl<<"After Sorting :"<<endl;**

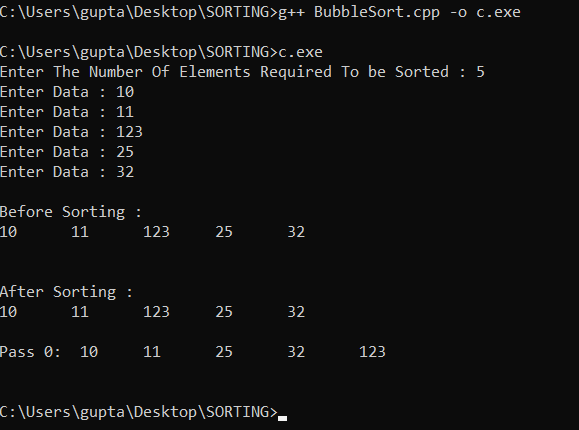
**b.ShowData();**

**b.Sort();**

**return 0;**

**}**

**Output**

****

**1.2 Quick\_Sort\_17.cpp**

**# include<iostream>**

**using namespace std;**

**class QuickSort**

**{**

**private:**

**int arr[100];**

**int noe,pivot,temp,s,e,i,j;**

**public:**

**void GetData()**

**{**

**cout<<"Enter size of array : "<<endl;**

**cin>>noe;**

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

**{**

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

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

**}**

**}**

**void ShowData()**

**{**

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

**{**

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

**}**

**cout<<endl;**

**}**

**void Sorted()**

**{**

**Quick(0,noe-1);**

**}**

**void Quick(int s,int e)**

**{**

**pivot=s;**

**i=pivot+1;**

**j=e;**

**if(j<i)**

**{**

**return;**

**}**

**while(arr[pivot]>arr[i])**

**{**

**i++;**

**if(j<i)**

**{**

**swap(arr[pivot],arr[j]);**

**pivot=j;**

**Quick(s,pivot-1);**

**Quick(pivot+1,e);**

**return;**

**}**

**}**

**while(arr[pivot]<arr[j])**

**{**

**j--;**

**if(j<i)**

**{**

**swap(arr[pivot],arr[j]);**

**pivot=j;**

**Quick(s,pivot-1);**

**Quick(pivot+1,e);**

**return;**

**}**

**}**

**swap(arr[i],arr[j]);**

**ShowData();**

**Quick(s,e);**

**return;**

**}**

**void swap(int &a,int &b)**

**{**

**temp=a;**

**a=b;**

**b=temp;**

**}**

**};**

**int main()**

**{**

**QuickSort d;**

**d.GetData();**

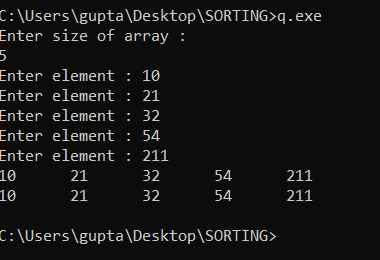
**d.ShowData();**

**d.Sorted();**

**d.ShowData();**

**}**

**Output**

****

**1.3 Selection\_Sort\_17.cpp**

**#include<iostream>**

**using namespace std;**

**class SelectionSort**

**{**

**private:**

**int noe,i,j,temp,index,count;**

**int arr[100];**

**public:**

**void GetData()**

**{**

**cout<<"Enter The Number Of Elements Required To be Sorted : ";**

**cin>>noe;**

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

**{**

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

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

**}**

**}**

**void ShowData()**

**{**

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

**{**

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

**}**

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

**}**

**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-1); i++)**

**{**

**if(Sorted())**

**break;**

**int min = arr[i];**

**for(j=(i+1); j<noe; j++)**

**{**

**if(min>arr[j])**

**{**

**min = arr[j];**

**count++;**

**index = j;**

**}**

**}**

**if(count!=0)**

**{**

**temp = arr[i];**

**arr[i] = min;**

**arr[index] = temp;**

**}**

**count=0;**

**cout<<"Pass "<<i<<" : ";**

**ShowData();**

**}**

**}**

**};**

**int main()**

**{**

**SelectionSort b=SelectionSort();**

**b.GetData();**

**cout<<endl<<"Before Sorting :"<<endl;**

**b.ShowData();**

**cout<<endl<<"After Sorting :"<<endl;**

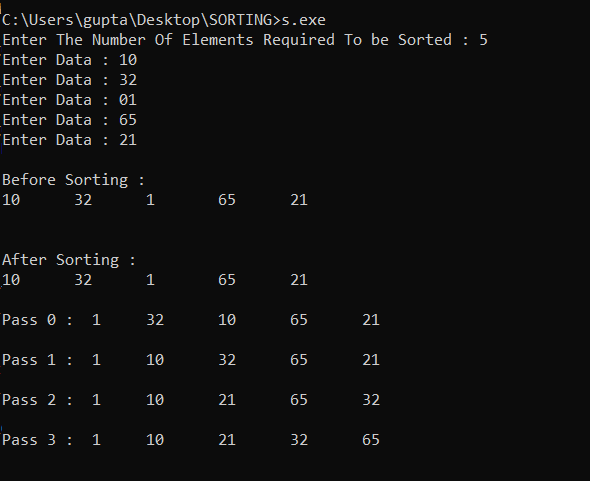
**b.ShowData();**

**b.Sort();**

**return 0;**

**}**

**Output**

****

**1.4 Radix\_Sort\_17.cpp**

**//Harshil’s Code :**

**#include<iostream>**

**using namespace std;**

**class Sort{**

**int arr[10];**

**int noe,size;**

**int bucket[10][20], buck\_count[10];**

**int i,j,k,r,no\_of\_passes,divisor,largest,pass\_no;**

**public:**

**Sort(int n,int s)**

**{**

**noe=n;**

**size=s;**

**no\_of\_passes=0;**

**divisor=1;**

**}**

**void getdata()**

**{**

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

**{**

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

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

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

**{**

**cout<<"negative number not allowed"<<endl;**

**cout<<"Re-Enter The Value :";**

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

**}**

**}**

**}**

**void sort()**

**{**

**cout<<"In sort\n";**

**largest=arr[0];**

**for(i=1;i<noe;i++)**

**{**

**if(Sorted())**

**{**

**cout<<"SORTED";**

**break;**

**}**

**if(arr[i] > largest)**

**largest=arr[i];**

**}**

**while(largest > 0)**

**{**

**no\_of\_passes++;**

**largest /= 10;**

**}**

**for(pass\_no=0; pass\_no < no\_of\_passes; pass\_no++)**

**{**

**for(k=0; k<10; k++)**

**buck\_count[k]=0;**

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

**{**

**r=(arr[i]/divisor) % 10;**

**bucket[r][buck\_count[r]++]=arr[i];**

**}**

**i=0;**

**for(k=0; k<10; k++)**

**{**

**for(j=0; j<buck\_count[k]; j++)**

**arr[i++] = bucket[k][j];**

**}**

**divisor =divisor \* 10;**

**showdata();**

**}**

**}**

**int Sorted()**

**{**

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

**{**

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

**return 0;**

**}**

**return 1;**

**}**

**void showdata()**

**{**

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

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

**cout<<endl;**

**}**

**};**

**int main(){**

**int noe, size;**

**cout<<"Enter size of array : ";**

**cin>>size;**

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

**cin>>noe;**

**if(noe>size)**

**cout<<"No of elements exceeds array size";**

**else**

**{**

**Sort s(noe,size);**

**s.getdata();**

**s.sort();**

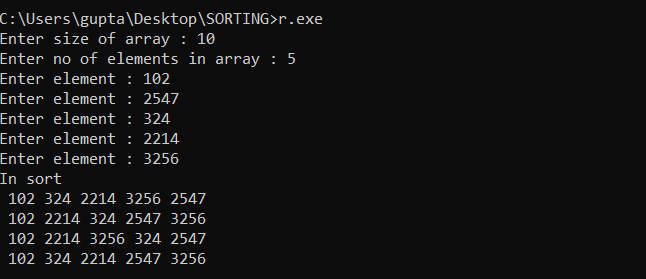
**}**

**finish:**

**return 0;**

**}**

**Output**

****

**1.5 Insertion\_Sort\_17.cpp**

**#include<iostream>**

**using namespace std;**

**class InsertionSort**

**{**

**private:**

**int noe,i,j,temp,index,count;**

**int arr[100];**

**public:**

**void GetData()**

**{**

**cout<<"Enter The Number Of Elements Required To be Sorted : ";**

**cin>>noe;**

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

**{**

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

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

**}**

**}**

**void ShowData(int num)**

**{**

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

**{**

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

**}**

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

**}**

**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;**

**temp=arr[i];**

**j=i;**

**while(j>0 && temp<arr[j-1])**

**{**

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

**j=j-1;**

**}**

**arr[j]=temp;**

**cout<<"Pass "<<i<<" : ";**

**ShowData(i+1);**

**}**

**}**

**};**

**int main()**

**{**

**InsertionSort b=InsertionSort();**

**b.GetData();**

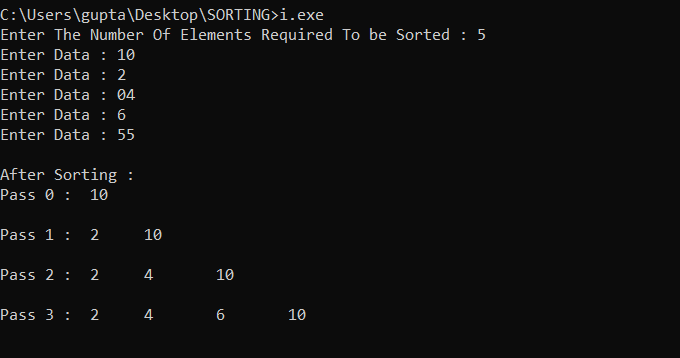
**cout<<endl<<"After Sorting :"<<endl;**

**b.Sort();**

**return 0;**

**}**

**Output**

****

**1.6 Shell\_Sort\_17.cpp**

**# include<iostream>**

**using namespace std;**

**class ShellSort**

**{**

**private:**

**int arr[100];**

**int noe,temp,i,j,increment,p;**

**public:**

**void GetData()**

**{**

**cout<<"Enter size of array : "<<endl;**

**cin>>noe;**

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

**{**

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

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

**}**

**}**

**void ShowData()**

**{**

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

**{**

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

**}**

**cout<<endl;**

**}**

**int Sorted()**

**{**

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

**{**

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

**return 0;**

**}**

**return 1;**

**}**

**void Sort()**

**{**

**for(increment = noe/2 ; increment > 0 ; increment=increment/2)**

**{**

**if(Sorted())**

**{**

**break;**

**}**

**for(j = increment ; j < noe ; j++)**

**{**

**for(i = j - increment ; i >= 0 ; i=i-increment)**

**{**

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

**{**

**break;**

**}**

**else**

**{**

**swap(arr[i+increment],arr[i]);**

**}**

**}**

**}**

**ShowData();**

**}**

**}**

**void swap(int &a,int &b)**

**{**

**temp=a;**

**a=b;**

**b=temp;**

**}**

**};**

**int main()**

**{**

**ShellSort d;**

**d.GetData();**

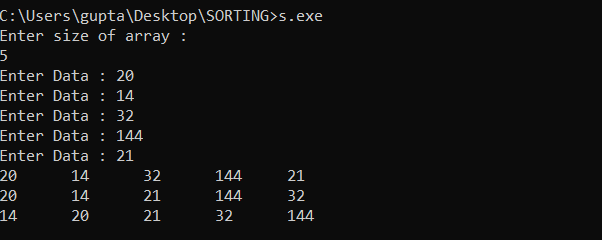
**d.ShowData();**

**d.Sort();**

**return 0;**

**}**

**Output**

****