**Code : Binary Search**

#include<stdio.h>

int binarySearch(int arr[],int size,int key);

void main(){

int size,key,indexFound=-1;

printf("enter array's size:");

scanf("%d",&size);

int arr[size];

printf("\nEnter %d elements in ascending order:\n",size);

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

scanf("%d",&arr[i]);

}

printf("\nEnter element to be searched: ");

scanf("%d",&key);

indexFound=binarySearch(arr,size,key);

if(indexFound>-1){

printf("%d found at index[%d]",key,indexFound);

}

else

printf("%d doesnot exist in the array",key);

}

**int binarySearch(int arr[],int size,int key)**{

if(size<1){

return -1;

}

int start=0;

int end=size-1;

int mid=0;

while(start<end){

mid=(start+end)/2;

if(key==arr[mid]){

return mid;

}

if(key>arr[mid]){

start=mid+1;

}

else{

end=mid-1;

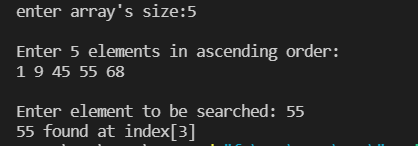
}

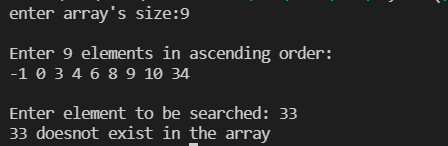
}

return -1;

}

Output: Binary Search





**Code : Insertion Sort**

#include<stdio.h>

void insertionSort(int arr[],int n);

void main(){

int size;

printf("enter array's size:");

scanf("%d",&size);

int arr[size];

printf("\nEnter %d elements:\n",size);

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

scanf("%d",&arr[i]);

}

insertionSort(arr,size);

printf("\nAfter applying InsertionSort:\n");

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

printf("%d ",arr[i]);

}

}

void insertionSort(int arr[],int n){

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

int j=i+1;

int key=arr[j];

while(j>0){

if(key<arr[j-1]){

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

j--;

}

else

break;

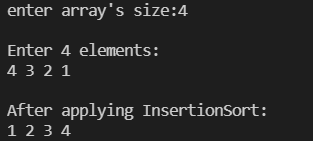
}

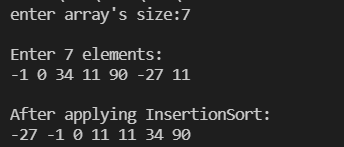
arr[j]=key;

}

}

Output: Insertion Sort





**Code : Quick Sort**

#include<stdio.h>

void quickSort(int arr[],int low,int high);

int partiton(int arr[],int low,int high);

void swap(int arr[],int index1,int index2);

void main(){

int size;

printf("enter array's size:");

scanf("%d",&size);

int arr[size];

printf("\nEnter %d elements:\n",size);

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

scanf("%d",&arr[i]);

}

quickSort(arr,0,size-1);

printf("\nAfter applying QuickSort:\n");

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

printf("%d ",arr[i]);

}

}

void quickSort(int arr[],int low,int high){

if(low<high){

int piv=partiton(arr,low,high);

quickSort(arr,low,piv-1);

quickSort(arr,piv+1,high);

}

}

int partiton(int arr[],int low,int high){

int pivot=arr[low];

int i=low;

int j=high;

while(i<j){

while(i<=high && arr[i]<=pivot){

i++;

}

while(arr[j]>pivot){

j--;

}

if(i<j){

swap(arr,i,j);

}

}

swap(arr,low,j);

return j;

}

void swap(int arr[],int index1,int index2){

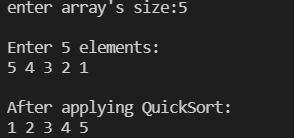
int temp=arr[index1];

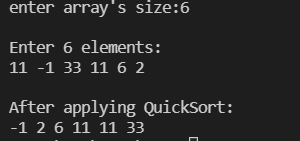
arr[index1]=arr[index2];

arr[index2]=temp;

}

OUTPUT: Quick Sort





**Code : Merge Sort**

#include<stdio.h>

void mergeSort(int arr[],int low, int high);

void merge(int arr[],int low,int mid,int high);

void main(){

int size;

printf("enter array's size:");

scanf("%d",&size);

int arr[size];

printf("\nEnter %d elements:\n",size);

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

scanf("%d",&arr[i]);

}

mergeSort(arr,0,size-1);

printf("\nAfter applying MergeSort:\n");

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

printf("%d ",arr[i]);

}

}

void mergeSort(int arr[],int low, int high){

if(low<high){

int mid= (low+high)/2;

mergeSort(arr,low,mid);

mergeSort(arr,mid+1,high);

merge(arr,low,mid,high);

}

}

void merge(int arr[],int low,int mid,int high){

int mixlen=high-low+1;

int mix[mixlen];

int i,j,k;

i=low;

j=mid+1;

k=0;

while(i<=mid && j<=high){

if(arr[i]<arr[j]){

mix[k]=arr[i];

i++;

}

else{

mix[k]=arr[j];

j++;

}

k++;

}

while(i<=mid){

mix[k]=arr[i];

i++;

k++;

}

while(j<=high){

mix[k]=arr[j];

j++;

k++;

}

for(int i=0;i<mixlen;i++){

arr[low++]=mix[i];

}

}

OUTPUT: Merge Sort

