QUICK SORT AND MERGE SORT

# Quick Sort:

**Code:**

#include<stdio.h>

int partition(int a[],int low,int high)

{

int i,j,temp,pivot; pivot=low; i=low+1;

j=high; while(i<=j)

{

while(a[i]<a[pivot])

{

i++;

}

while(a[j]>a[pivot])

{

j--;

}

if(i<j)

{

temp=a[i]; a[i]=a[j]; a[j]=temp;

}

}

temp=a[low]; a[low]=a[j]; a[j]=temp; return j;

}

void qsort(int a[],int low,int high)

{

int mid; if(low<high)

{

mid=partition(a,low,high); qsort(a,low,mid-1); qsort(a,mid+1,high);

}

}

void main()

{

int a[100],n,i,low,high;

printf("\nEnter the number of elements:"); scanf("%d",&n);

printf("\nEnter the elements:"); for(i=0;i<n;i++)

{

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

}

low=0; high=n-1;

qsort(a,low,high);

printf("\nSorted Elements are:\t"); for(i=0;i<n;i++)

{

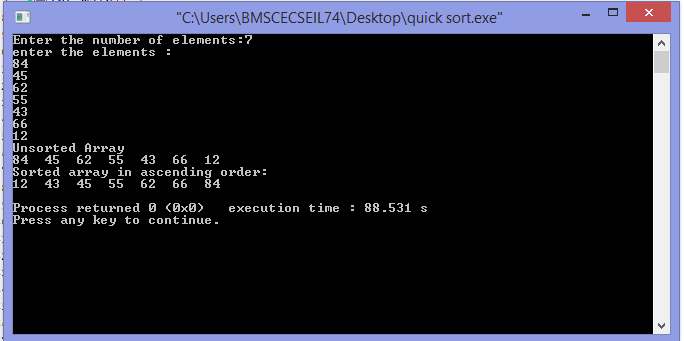
printf("%d\t",a[i]);

}

getch();

}

# Output:



**Merge Sort:**

# Code:

#include<stdio.h>

void merge(int a[],int low,int mid,int high)

{

int i,j,k,c[100]; i=low; j=mid+1; k=low;

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

{

if(a[i]<a[j])

{

c[k++]=a[i++];

}

else

{

c[k++]=a[j++];

}

}

while(i<=mid)

{

c[k++]=a[i++];

}

while(j<=high)

{

c[k++]=a[j++];

}

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

{

a[i]=c[i];

}

}//merge

void mergesort(int a[],int low,int high)

{

int mid; if(low<high)

{

mid=(low+high)/2; mergesort(a,low,mid); mergesort(a,mid+1,high); merge(a,low,mid,high);

}

}//mergeSort

void main()

{

int a[100],n,i,low,high;

printf("\nEnter the number of elements to be sorted:"); scanf("%d",&n);

printf("\nEnter the elements:"); for(i=0;i<n;i++)

{

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

}

low=0; high=n-1;

printf("Unsorted elements :\t"); for(i=0;i<n;i++)

{

printf("%d\t",a[i]);

}

mergesort(a,low,high); printf("\nSorted Elements are: \t"); for(i=0;i<n;i++)

{

printf("%d\t",a[i]);

}

//main

# Output:

