Program 1

```
#include<stdio.h> //Merge Sort time complexity is (n*log(n)) where is log
is base 2
#include<stdlib.h>
#define MAX 1000
int count;
void merge(int a[MAX], int low, int mid, int high) //merges in ascending
order
{
int i, j, k, b[MAX]; //b is temp array to store the sorted elements
i = low;
j = mid+1;
k = low;
while(i<=mid && j<=high)
{
if(a[i]<a[j])
{
b[k++] = a[i++];
}
else
{
b[k++] = a[j++];
}
count++;
}
while(i<=mid)
{
b[k++] = a[i++];
count++;
}
while(j<=high)
{
b[k++] = a[j++];
count++;
}
for(i=low;i<=high;i++)
a[i] = b[i]; //the sorted elements are again put back into initial array a
only
}
void mergesort(int a[MAX], int low, int high) //splits the array into two
```

```
halves
{
int mid;
if(low<high)
{
int mid = (low+high)/2;
mergesort(a, low, mid);
mergesort(a, mid+1, high);
merge(a, low, mid, high);
}
}
int main()
{
int i, j, n, a[MAX], b[MAX], c[MAX];
int c1, c2, c3;
printf("\nMERGE SORT\n");
printf("\nEnter the number of elements in the array - ");
scanf("%d",&n);
printf("\nEnter the elements of the array - ");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
count=0;
mergesort(a,0,n-1);
printf("\nThe sorted elements of the array - ");
for(i=0;i<n;i++)
printf("%d ",a[i]);
printf("\n\nThe number of counts- %d\n",count); //prints the number of
times the
//basic operation is performed(comparision) is for this array.
/* TIME COMPLEXITY ANALYSIS OF MERGE SORT*/
printf("\nSIZE\tASC\tDESC\tRAND\n"); //for time complexity analysis, using
3

for(i=16;i<550;i=i*2){
for(j=0;j<i;j++){
a[j]=j; //array is filled with elements in strictly ascending order

b[j]=i-j; //array is filled with elements in strictly descending order -->

c[j]=rand()%i; //array is filled with elements in randomn order

}
count = 0;
```

```
mergesort(a,0,i-1); //ascending array is sorted, and number of basic

c1 = count;
count = 0;
mergesort(b,0,i-1); //descending array is sorted, and number of basic

c2 = count;
count = 0;
mergesort(c,0,i-1); //randomn array is sorted, and number of basic
operations

c3 = count;
printf("\n %d\t %d\t %d\t %d",i, c1, c2, c3);
}
printf("\n");
return(0);
}
```