

NAME:	Pranav Sadanand Kore
UID:	2021300065
SUBJECT	Design and Analysis of Algorithm
EXP No.	2
AIM:	Sorting the Array using Merge Sort Algorithm
ALGORITHM:	<ul style="list-style-type: none"> MergeSort function if left < right mid= (left+right)/2 MergeSort(array, left, mid) MergeSort (array, mid+1, right) Merge(array, left, mid, right) Merge function n1= mid-left+1 n2= right-q create arrays L[n1+1] and R[n2+1] for i from 0 to n1-1 do L[i]=array[p+i] for i from 0 to n2-1 do R[i]=array[q+i+1] L[n1]= aprox. Infinity R[n2]= aprox. Infinity Declare k, i=1 and j=1 For k from left to right Do if L[i]<=R[j] Then array[k]=L[i] i++ else array[k]=R[j] j++

PROGRAM:

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>

void Printarr(int arr[],int n)
{
    for(int i=0; i<n; i++)
    {
        printf("%d ",arr[i]);
    }
    printf("\n");
}

void Merge(int arr[],int p,int q,int r)
{
    int n1,n2;
    n1=q-p+1;
    n2=r-q;
    int L[n1+1], R[n2+1];
    for(int i=0; i<n1; i++)
    {
        L[i]=arr[p+i];
    }
    for(int i=0; i<n2; i++)
    {
        R[i]=arr[q+i+1];
    }
    L[n1]=99999999;
    R[n2]=99999999;

    int i=0,j=0;
    for(int k = p ; k <= r ; k++)
    {
        if(L[i] <= R[j])
        {
            arr[k] = L[i];
            i++;
        }
        else
        {
            arr[k] = R[j];
            j++;
        }
    }
}
```

```

    }
}
return;
}

void MergeSort(int arr[],int p,int r)
{
    int q;
    if(r>p)
    {
        int q = (p+r)/2;
        MergeSort(arr,p,q);
        MergeSort(arr,q+1,r);
        Merge(arr,p,q,r);
    }
    return;
}

int main()
{
    int n;
    printf("Enter the no. of elements in the array:");
    scanf("%d",&n);
    int arr[n];

    printf("Enter elements in array:\n");
    for(int i=0; i<n; i++)
    {
        scanf("%d",&arr[i]);
    }

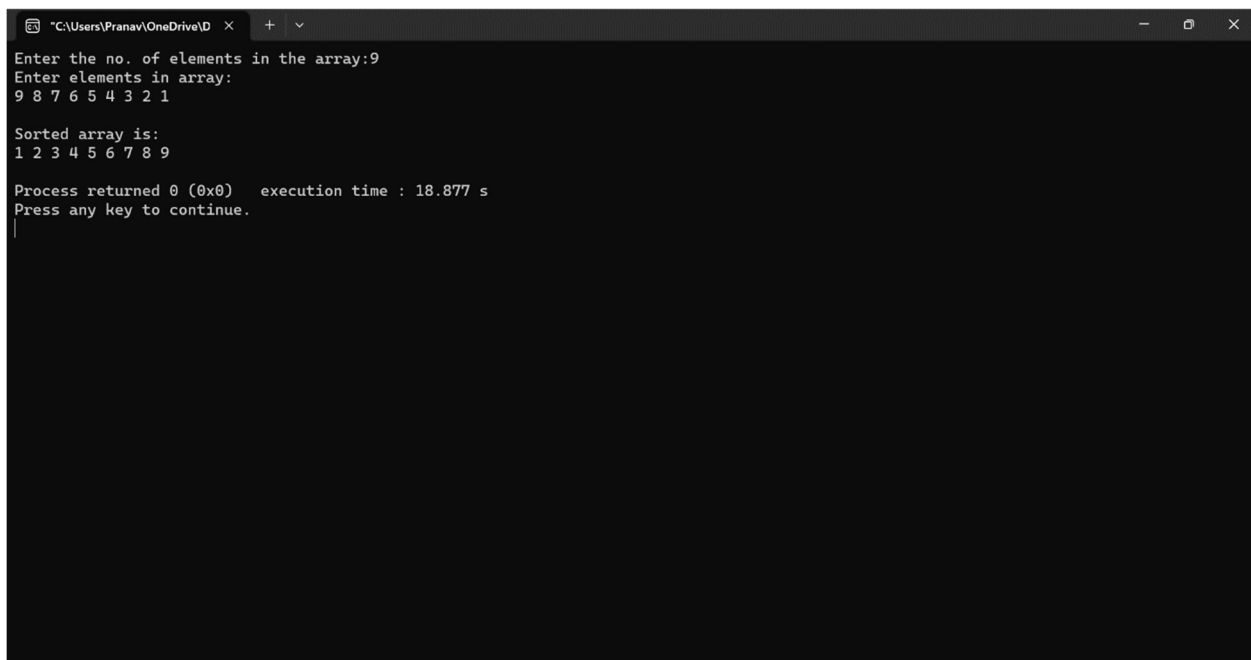
    MergeSort(arr,0,n-1);

    printf("\nSorted array is:\n");
    Printarr(arr,n);

    return 0;
}

```

RESULT (SNAPSHOT)

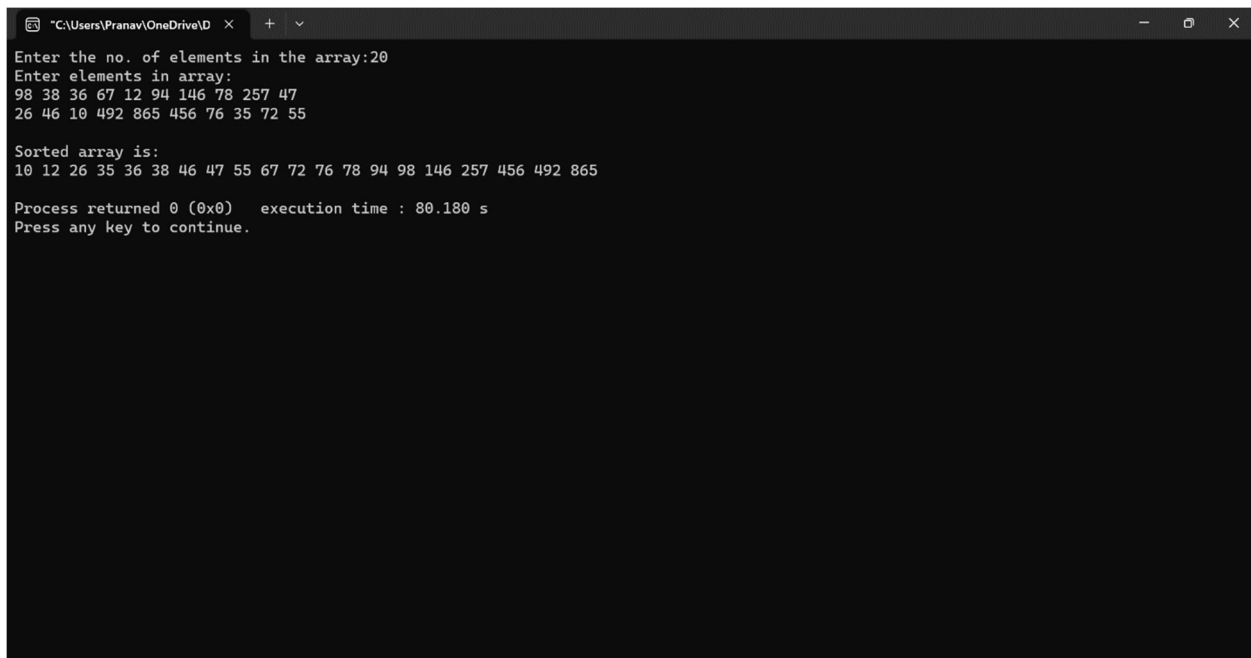


A screenshot of a Windows command prompt window titled "C:\Users\Pranav\OneDrive\ID". The window shows the execution of a C++ program. The user is prompted to enter the number of elements in the array, which is 9. Then, the user is prompted to enter the elements in the array, which are 9 8 7 6 5 4 3 2 1. The program then displays the sorted array as 1 2 3 4 5 6 7 8 9. Finally, it shows the process returned 0 (0x0) and the execution time was 18.877 s. The prompt "Press any key to continue." is shown at the bottom.

```
"C:\Users\Pranav\OneDrive\ID" x + v
Enter the no. of elements in the array:9
Enter elements in array:
9 8 7 6 5 4 3 2 1

Sorted array is:
1 2 3 4 5 6 7 8 9

Process returned 0 (0x0)   execution time : 18.877 s
Press any key to continue.
|
```



A screenshot of a Windows command prompt window titled "C:\Users\Pranav\OneDrive\ID". The window shows the execution of a C++ program. The user is prompted to enter the number of elements in the array, which is 20. Then, the user is prompted to enter the elements in the array, which are 98 38 36 67 12 94 146 78 257 47 26 46 10 492 865 456 76 35 72 55. The program then displays the sorted array as 10 12 26 35 36 38 46 47 55 67 72 76 78 94 98 146 257 456 492 865. Finally, it shows the process returned 0 (0x0) and the execution time was 80.180 s. The prompt "Press any key to continue." is shown at the bottom.

```
"C:\Users\Pranav\OneDrive\ID" x + v
Enter the no. of elements in the array:20
Enter elements in array:
98 38 36 67 12 94 146 78 257 47
26 46 10 492 865 456 76 35 72 55

Sorted array is:
10 12 26 35 36 38 46 47 55 67 72 76 78 94 98 146 257 456 492 865

Process returned 0 (0x0)   execution time : 80.180 s
Press any key to continue.
```

CONCLUSION:	The experiment demonstrated the efficiency and scalability of the merge sort algorithm. The results showed that the algorithm had a linear time complexity of $O(n \log n)$, making it an ideal choice for sorting large data sets. The results were consistent with the expected results, and the algorithm was able to sort the data sets efficiently and accurately.
--------------------	--