PRACTICAL 5

AIM: Write a program to sort given elements of an array in ascending order using Merge sort. Analyze the time complexity for best, average and worst case.

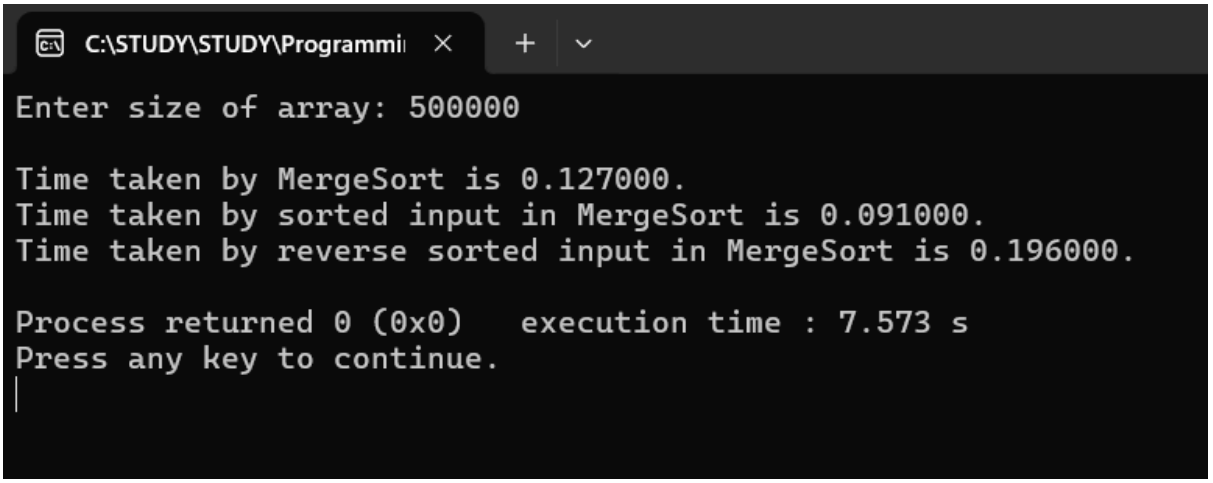
```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <limits.h>

void MergeSort(int[], int , int );
void Merge(int[], int, int, int );


void reverseArray(int* arr, int size) {
   int start = 0;
   int end = size - 1;
   while (start < end) {
      int temp = arr[start];
      arr[start] = arr[end];
      arr[end] = temp;
      start++;
      end--;
   }
}

int main()
{
   int SIZE;
   printf("Enter size of array: ");
   scanf("%d",&SIZE);
   int *arr = (int*)malloc(SIZE*sizeof(int));

   for(int i = 0;i<SIZE;i++)
   {
      arr[i] = rand() % SIZE;
   }

   int p = 0;
```

```
int r = SIZE - 1;

clock_t start,end;
double cpu_time;

start = clock();

MergeSort(arr,p,r);

end = clock();
/*
for(int t = 0 ;t<SIZE;t++){
   printf("%d ",arr[t]);
}
*/
printf("\n");

cpu_time = ((double)end - start)/CLOCKS_PER_SEC;
printf("Time taken by MergeSort is %f.\n",cpu_time);

clock_t start2,end2;
double cpu_time2;

start2 = clock();

MergeSort(arr,p,r);

end2 = clock();

cpu_time2 = ((double)end2 - start2)/CLOCKS_PER_SEC;
printf("Time taken by sorted input in MergeSort is %f.\n",cpu_time2);



reverseArray(arr , SIZE) ;
clock_t start3,end3;
double cpu_time3;

start2 = clock();
MergeSort(arr,p,r);
end2 = clock();

cpu_time3 = ((double)end3 - start3)/CLOCKS_PER_SEC;
printf("Time taken by reverse sorted input in MergeSort is %f.\n",cpu_time3);
```

```
    free(arr);
    return 0;
}

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

void Merge(int A[], int p, int q, int r)
{
    int n1 = q - p + 1;
    int n2 = r - q;
    int* L = (int*)malloc((n1 + 1)*sizeof(int));
    int* R = (int*)malloc((n2 + 1)*sizeof(int));
    for(int i = 0;i<n1;i++)
    {
        L[i] = A[p+i];
    }
    for(int j = 0;j<n2;j++)
    {
        R[j] = A[q+j+1];
    }
    L[n1] = INT_MAX;
    R[n2] = INT_MAX;
    int i = 0;
    int j = 0;
    for(int k = p;k<=r;k++)
    {
        if(L[i] <= R[j]){
            A[k] = L[i];
            i = i + 1;
        }
        else{
```

```
            A[k] = R[j];
            j = j + 1;
        }
    }

    free(L);
    free(R);
}
```

OUTPUT:



Output for different size of input

	150000	250000	500000	750000	1000000
Random Array	0.03700	0.06300	0.12700	0.19100	0.25700
Sorted Array	0.03000	0.04500	0.09100	0.14100	0.18900
Reverse Sorted Array	0.19600	0.19600	0.19600	0.19600	0.19600

Where elapsed time is in seconds.