AOA PRACTICAL LAB 1

Name: Brendan Lucas, Roll No:8953, Div: SE Comp B

Source Code:

```
#include <iostream>
#include <cstdlib>
using namespace std;

void mergearray(int *arr,int l,int m,int r)
{
	//Declaring variables
	int j,i,num,al1,al2,*temparr1,*temparr2;

	//length of array 1
	al1=m-l+1;

	//length of array 2
	al2=r-m;

	//create array 1 using malloc
	temparr1=(int*)malloc(sizeof(int)*al1);

	//create array 2 using malloc
	temparr2=(int*)malloc(sizeof(int)*al2);

	//copy all elements of partition into temp array
	for(i=0;i<al1;i++)
	{
		temparr1[i]=arr[l+i];
	}
	for(i=0;i<al2;i++)
	{
		temparr2[i]=arr[m+1+i];
	}

	//Set counters for sorting
	i=0;
	j=0;
	num=l;

	//Till both arrays are not exhusted
	while(i<al1&&j<al2)
	{
		//If first array element has smaller value
		if(temparr1[i]<temparr2[j])
		{
			arr[num]=temparr1[i];
			i++;
		}
		//If second array value has smaller value
		else
		{
			arr[num]=temparr2[j];
			j++;
		}
		num++;
	}

	//if array 2 is exhausted then copy all elements of array 1
	while(i<al1)
	{
		arr[num]=temparr1[i];
		i++;
		num++;
	}
	//if array 1 is exhausted then copy all elements of array 2
	while(j<al2)
	{
		arr[num]=temparr2[j];
		j++;
		num++;
	}

	//release the space of temp arrays
```

```
	free(temparr1);
	free(temparr2);
}

void mergesort(int *arr,int l,int r)
{
	//Declaring variables
	int m;

	//if partition has one or more than one elements
	if(r>l)
	{
		//Find approx middle of array
		m=(r+l)/2;
		//Recursive call for first partition
		mergesort(arr,l,m);
		//Recursive call for second partition
		mergesort(arr,m+1,r);
		//Merge the two partitions
		mergearray(arr, l, m, r);
	}

}

int main(void)
{

	//Declering variables
	int n,*arr,i;

	//Asking for No. of elements in array
	cout<<"Enter the no of elements in array:-\n";
	cin>>n;

	//Creating array of given size using malloc
	arr=(int*)malloc(sizeof(int)*n);

	//Taking input of elements of array
	cout<<"Enter the array\n";
	for(i=0;i<n;i++)
	{
		cin>>arr[i];
	}

	//Sorting the array
	//bubblesort(arr, n);
	//insertionsort(arr, n);
	mergesort(arr,0, n-1);

	//Displaying the final result
	cout<<"\nThe Sorted array is:- \n";
	for(i=0;i<n;i++)
	{
		cout<<arr[i]<<" ";
	}
	return 0;
}
```

Merge Sort :-

code :-

```
void mergearray(int *arr, int l, int m, int r)
{
    al1 = m - l + 1;
    al2 = r - m;
    temparr1 = (int*)malloc(sizeof(int)*al1);
    temparr2 = (int*)malloc(sizeof(int)*al2);
    for(i=0; i<al1; i++)
    {
        temparr1[i] = arr[l+i];
    }
    for(i=0; i<al2; i++)
    {
        temparr2[i] = arr[m+l+i];
    }
    i=0;
    j=0;
    num=l;
    while(i<al1 && j<al2)
    {
        if(temparr1[i] < temparr2[j])
        {
            arr[num] = temparr1[i];
            i++;
        }
        else{
            arr[num] = temparr2[j];
            j++
        }
        num++
    }
```

```
while(i<al1)
{
    arr[num]=temparr[i];
    i++;
    num++;
}
while(j<al2)
{
    arr[num] = temparr[j];
    j++;
    num++;
}
free(temparr1);
free(temparr2);
}

void mergesort(int *arr, int l, int r)
{
    int m
    if(r>l)
    {
        m=(r+l)/2;
        mergesort(arr, l, m);
        mergesort(arr, m+1, r);
        mergearray(arr, l, m, r);
    }
}
```

Page No.
Date

Time analysis :-

$T(n) = 2T\left(\frac{n}{2}\right) + n$ [Time required to sort the array].

Here, $a = 2$ & $b = 2$.

By master method

$$n^{\log_b a} = n^{\log_2 2} = n^1 = n.$$

$f(n) = n$

$f(n) = n^{\log_b a}$

Using results of case 2 in master method

$$T(n) = \Theta(n \log_2 n).$$