Motilal Nehru National Institute of Technology Allahabad, Prayagraj
Computer Science & Engineering Department
Analysis of Algorithm Lab

Assignment-2

SHISHU

2020CA089

Q-1: Write a C Program to analyse the complexity of Merge Sort Algorithm. Also plot its graph for all cases.

```c
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#define size 100000000

void mergeSort(long int a[],long int ,long int);
void merge(long int a[],long int,long int,long int);

void mergeSort(long int arr[],long int l,long int r)
    {
        if(l < r){
            int mid = l + (r - l)/2;
            mergeSort(arr, l, mid);
            mergeSort(arr, mid + 1, r);

            merge(arr, l, mid, r);
        }
    }

void merge(long int arr[],long int l,long int m,long int r)
    {
         int part1 = m - l + 1;
         int part2 = r - m;
         int tmp1[part1];
         int tmp2[part2];

         for(int i = 0; i < part1; i++)
            tmp1[i] = arr[l + i];
         for(int i = 0; i < part2; i++)
            tmp2[i] = arr[m + 1 + i];

         int i = 0, j = 0;
         int k = l;
```

```
        while(i < part1 && j < part2){
            if(tmp1[i] <= tmp2[j])
               arr[k++] = tmp1[i++];
            else
               arr[k++] = tmp2[j++];
        }

        while(i < part1)
            arr[k++] = tmp1[i++];

        while(j < part2)
            arr[k++] = tmp2[j++];


    }
int main(){
    FILE *fp;

    long int n=10000;
    int it=0;

    //arrays to store time duration
    //of sorting algorithms
    double time1[10];

    fp=fopen("mergeSort.txt","w");

    //fprintf(fp,"ArraySize  ExecutionTime\n");
    printf("ArraySize  ExecutionTime\n");

    //performs 5 Iterations
    while(it++<5){
        long int a[n];//mergeSort Array
        //generating n random numbers
        //storing them in arrays a;
        for(int i=0;i<n;i++){
            long int no= rand()%n+1;
            a[i]=no;
        }

        //using clock_t to tore time
        clock_t start,end;
        //mergeSort
        start=clock();

        mergeSort(a,0,n-1);
        end=clock();
```

```
        time1[it]=((double)(end-start));

        //type conversion to long int for
        //plotting graph whit integer values
       // fprintf("%li\t\t%li\n",n,(long int)time1[it]);
        printf("%li\t\t%li\n",n,(long int)time1[it]);
        n+=10000;

    }
   fclose(fp);

    return 0;
}
```

mergeSort.txt

ArraySize ExecutionTime

10000 4

20000 7

30000 7

40000 9

50000 10

dataplot.p

```
  set autoscale                 # scale axes automatically
    unset log                     # remove any log-scaling
    unset label                   # remove any previous labels
    set xtic auto                 # set xtics automatically
    set ytic auto                 # set ytics automatically
        set tics font "Helvetica,10"
    set title "Calculate Time Comlexity"
```

set xlabel "Input Size"

set ylabel "Time Taken"

#set key 0.01,100

#set label "Yield Point" at 0.003,260

#set arrow from 0.0028,250 to 0.003,280

set xr [1000:200000]

set yr [0.000000:50]

plot "mergeSort.txt" using 1:2 title 'MergeSort' with linespoints