SATYAM TRIPATHI
202151141

LAB 8

CODE

```
import java.util.Arrays;

class Sorting{
    //Quick Sort

    public Long QuickSortTime(int arr[],int n){
        Long start=System.nanoTime();
        QuickSort(arr, 0,n-1);
        Long end=System.nanoTime();
        return end-start;
    }
    public void QuickSort(int arr[],int l,int r){
        if(l<r){
            int pi=partition(arr,l,r);
            QuickSort(arr, l, pi-1);
            QuickSort(arr, pi+1, r);
        }
    }
    public void swap(int arr[],int i,int j){
        int temp=arr[i];
        arr[i]=arr[j];
        arr[j]=temp;
    }
    public int partition(int arr[],int l,int r){
        int pivot=arr[r];
        int i=l-1;
        for(int j=l;j<r;j++)
        {
            if(arr[j]<pivot){
                i++;
                swap(arr,i,j);
            }
```

SATYAM TRIPATHI
202151141

```
        }
        swap(arr,i+1,r);
        return i+1;
    }


    //Merge Sort

    public long MergeSortTime(int arr[],int n){
        long start=System.nanoTime();
        Mergesort(arr, 0, n-1);
        long end=System.nanoTime();
        return end-start;

    }
    public void Mergesort(int arr[],int l,int r){
        int mid=l+(r-l)/2;
        if(l<r){
            Mergesort(arr,l,mid);
            Mergesort(arr,mid+1,r);
            merge(arr,l,mid,r);
        }
    }
    public void merge(int arr[],int l,int mid,int r){
        int n1=mid-l+1;
        int n2=r-mid;
        int arr1[]=new int[n1];
        int arr2[]=new int[n2];
        for(int i=0;i<n1;i++)
        {
            arr1[i]=arr[l+i];
        }
        for(int i=0;i<n2;i++)
        {
            arr2[i]=arr[mid+i+1];
        }
        int i=0,j=0,k=0;
        while(i<n1 && j<n2){
```

SATYAM TRIPATHI
202151141

```
            if(arr1[i]<=arr2[j]){
                arr[l+k]=arr1[i];
                i++;
            }
            else{
                arr[l+k]=arr2[j];
                j++;
            }
            k++;
        }
        while(i<n1){
            arr[l+k]=arr1[i];
            i++;
            k++;
        }
        while(j<n2){
            arr[l+k]=arr2[j];
            j++;
            k++;
        }

    }
}
public class SortingAlgos {
    public static void main(String[] args) {
        double avgRandom=0.0,avgAscend=0.0,avgDescend=0.0;
        Sorting x1=new Sorting();
        for(int i=0;i<10;i++)
        {
            int arr[]=new int[100];
            for(int j=0;j<100;j++)
            {
                arr[j]=(int)(100.0*(double)Math.random());
            }
            //System.out.println(Arrays.toString(arr));
            avgRandom+=x1.MergeSortTime(arr, 100);
            Arrays.sort(arr);
            //System.out.println(Arrays.toString(arr));
            avgAscend+=x1.MergeSortTime(arr, 100);
```

SATYAM TRIPATHI
202151141

```
            for(int j=0;j<50;j++){
                int temp=arr[j];
                arr[j]=arr[99-j];
                arr[99-j]=temp;
            }
            //System.out.println(Arrays.toString(arr));
            avgDescend+=x1.MergeSortTime(arr, 100);

        }
        avgRandom/=10;
        avgRandom/=1000000;
        avgAscend/=10;
        avgAscend/=1000000;
        avgDescend/=10;
        avgDescend/=1000000;
        System.out.println("Merge sort Random Order = "+avgRandom);
        System.out.println("Merge sort Ascending Order = "+avgAscend);
        System.out.println("Merge sort Descending Order = "+avgDescend);

    //Quick Sort
    avgAscend=0.0;
    avgDescend=0.0;
    avgRandom=0.0;
    for(int i=0;i<10;i++)
    {
        int arr[]=new int[100];
        for(int j=0;j<100;j++)
        {
            arr[j]=(int)(100.0*(double)Math.random());
        }
        //System.out.println(Arrays.toString(arr));
        avgRandom+=x1.QuickSortTime(arr, 100);
        Arrays.sort(arr);
        //System.out.println(Arrays.toString(arr));
        avgAscend+=x1.QuickSortTime(arr, 100);
        for(int j=0;j<50;j++){
            int temp=arr[j];
            arr[j]=arr[99-j];
            arr[99-j]=temp;
```

SATYAM TRIPATHI
202151141

```
        }
        //System.out.println(Arrays.toString(arr));
        avgDescend+=x1.QuickSortTime(arr, 100);

    }
    avgRandom/=10;
    avgRandom/=1000000;
    avgAscend/=10;
    avgAscend/=1000000;
    avgDescend/=10;
    avgDescend/=1000000;
    System.out.println("Quick sort Random Order = "+avgRandom);
    System.out.println("Quick sort Ascending Order = "+avgAscend);
    System.out.println("Quick sort Descending Order = "+avgDescend);


    }
}
```

OUTPUT

```
Merge sort Random Order = 0.04088
Merge sort Ascending Order = 0.03225
Merge sort Descending Order = 0.03188
Quick sort Random Order = 0.02861
Quick sort Ascending Order = 0.09957
Quick sort Descending Order = 0.07813
```

SATYAM TRIPATHI
202151141

TABLE

S No.	Sorting Algorithm	Theoritical Time Complexity	Number of elements in array	Avg Time (in ms) Random	Avg Time (in ms) Ascending	Avg Time (in ms) Descending
1	Quick Sort	O(n log n)	100	0.02861	0.09957	0.07813
2	Merge Sort	O(n log n)	100	0.04088	0.03225	0.03188