Experiment No. 5
Sort a given set of n integer elements using Merge Sort method and compute its time complexity. Run the program for varied values of n > 5000, and record the time taken to sort. Plot a graph of the time taken versus n on graph sheet. The elements can be read from a fileor can be generated using the random number generator. Demonstrate using Java how the divide and- conquer method works along with its time complexity analysis: worst case,average case and best case.

```
import java.util.*;
public class mergesort {
	static int smax= 100;
	public void merge(int a[],int low,int mid,int high)
	{
		int b[]=new int[smax];
		int i,j,k;
		i=low;
		j=mid+1;
		k=low;
		while(i<=mid &&j<=high)
		{
			if(a[i]<a[j])
			{
				b[k]=a[i];
				i++;
				k++;
			}
			else
			{
				b[k]=a[j];
				j++;
				k++;
			}
		}
		while(i<=mid)
		{
			b[k]=a[i];
			k++;
			i++;
		}
		while(j<=high)
		{
			b[k]=a[j];
			k++;
			j++;
		}
		for(int m=low;m<=high;m++)
			a[m]=b[m];
	}
	void sort(int a[],int low,int high)
	{
		if(low<high)
		{
			int mid= (low+high)/2;
			sort(a,low,mid);
			sort(a,mid+1,high);
			merge(a,low,mid,high);
		}
	}
	public static void main(String[] args) {
		Scanner sc= new Scanner(System.in);
		System.out.println("Enter the size of the array");
		int n= sc.nextInt();
```

```
            int a[]= new int[n];
            Random rand= new Random();
            for(int i=0;i<n;i++)
                  a[i]= rand.nextInt(30);
            System.out.println("Before Sorting");
            for(int i=0;i<n;i++)
                  System.out.println(a[i]);
            long start_time= System.nanoTime();
            mergesort m= new mergesort();
            m.sort(a,0,n-1);
            long stop_time= System.nanoTime();
            long elapse_time= (stop_time-start_time);
            System.out.println("time taken to sort the array is "+
elapse_time+"nano seconds");
            System.out.println("the sorted array is ");
            for(int i=0;i<n;i++)
                  System.out.println(a[i]);

      }
}
```