2. Implement a merge sort algorithm to sort a given set of elements and determine the time required to sort the elements. Repeat the experiment for different values of n, the number of elements in the list to be sorted and plot a graph of the time taken versus n. The elements can be read from a file or can be generated using the random number generator.

```
import java.util.Random;
import  java.util.Scanner;
public class Mergesort
{
	static final int MAX = 10005;
	static int[] a = new int[MAX];

	public static void main(String[] args)
	{
		Scanner input = new Scanner(System.in);
		System.out.print("Enter Max array size: ");
		int n = input.nextInt();
		Random random = new Random();
		System.out.println("Enter the array elements: ");
		for (int i = 0; i < n; i++)
			a[i] = random.nextInt(1000);
		for (int i = 0; i < n; i++)
			System.out.print(a[i] + " ");
		long startTime = System.nanoTime();
		MergeSortAlgo(0, n - 1);
		long stopTime = System.nanoTime();
		long elapsedTime = stopTime - startTime;
		System.out.println("Time Complexity (ms) for n = " + n + " is : " +
		(double)elapsedTime / 1000000);
		System.out.println("Sorted Array (Merge Sort):");
		for (int i = 0; i < n; i++)
			System.out.print(a[i] + " ");
		input.close();
	}
	public static void MergeSortAlgo(int low, int high)
		{
			int mid;
			if (low < high)
			{
				mid = (low + high) / 2;
				MergeSortAlgo(low, mid);
				MergeSortAlgo(mid + 1, high);
				Merge(low, mid, high);
			}
		}

		public static void Merge(int low, int mid, int high)
		{
```

```
                int[] b = new int[MAX];
                int i, h, j, k;
                h = i = low;
                j = mid + 1;
                while ((h <= mid) && (j <= high))
                        if (a[h] < a[j])
                                b[i++] = a[h++];
                        else
                                b[i++] = a[j++];

                if (h > mid)
                        for (k = j; k <= high; k++)
                                b[i++] = a[k];
                else
                        for (k = h; k <= mid; k++)
                                b[i++] = a[k];

                for (k = low; k <= high; k++)
                        a[k] = b[k];
        }
}
```