CS162

ASSIGNMENT 8

NAME:

ARCHIT AGRAWAL

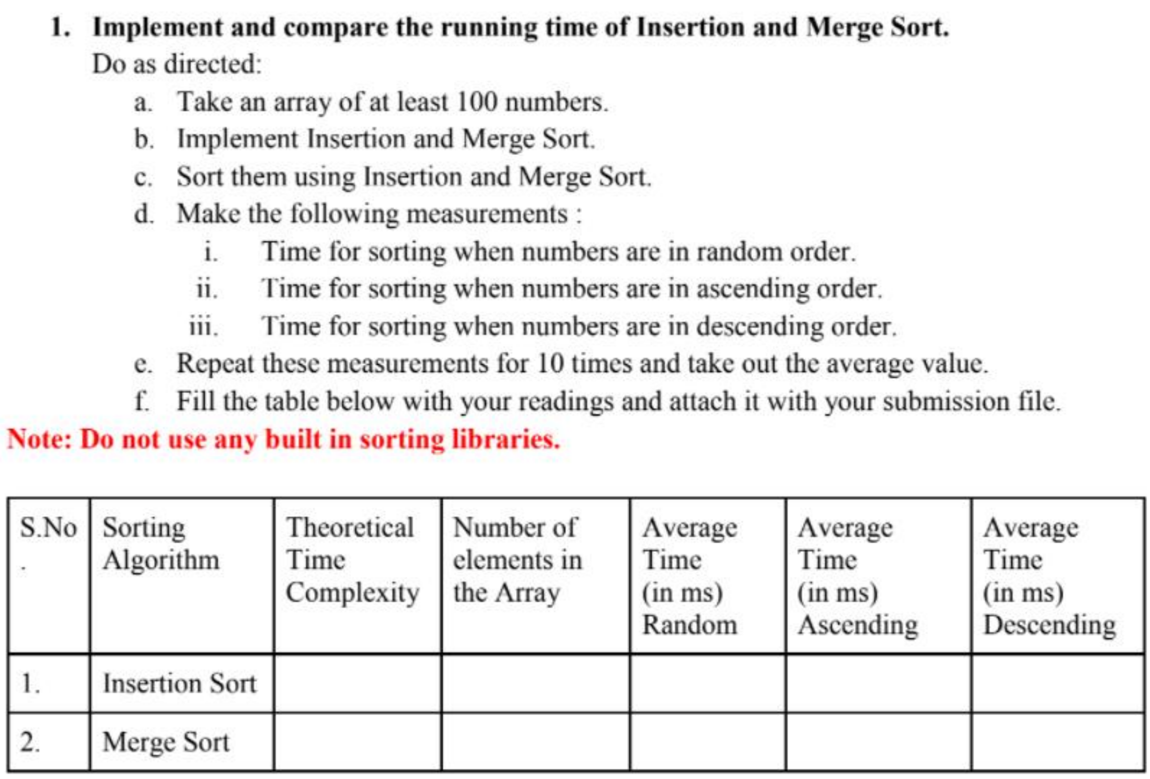
ROLL NO. :

202052307

SECTION:

A

Question



1. **Implement and compare the running time of Insertion and Merge Sort.**
   Do as directed:
   a. Take an array of at least 100 numbers.
   b. Implement Insertion and Merge Sort.
   c. Sort them using Insertion and Merge Sort.
   d. Make the following measurements :
      i. Time for sorting when numbers are in random order.
      ii. Time for sorting when numbers are in ascending order.
      iii. Time for sorting when numbers are in descending order.
   e. Repeat these measurements for 10 times and take out the average value.
   f. Fill the table below with your readings and attach it with your submission file.

Note: Do not use any built in sorting libraries.

S.No.	Sorting Algorithm	Theoretical Time Complexity	Number of elements in the Array	Average Time (in ms) Random	Average Time (in ms) Ascending	Average Time (in ms) Descending
1.	Insertion Sort					
2.	Merge Sort					

Description -: The following program asks the user to input the size of array and asks to decide in what order(ascending, descending or random) the user wants to pass the array to sorting methods (insertion sort, merge sort).

The array is generated using random function, hence it is not sorted.

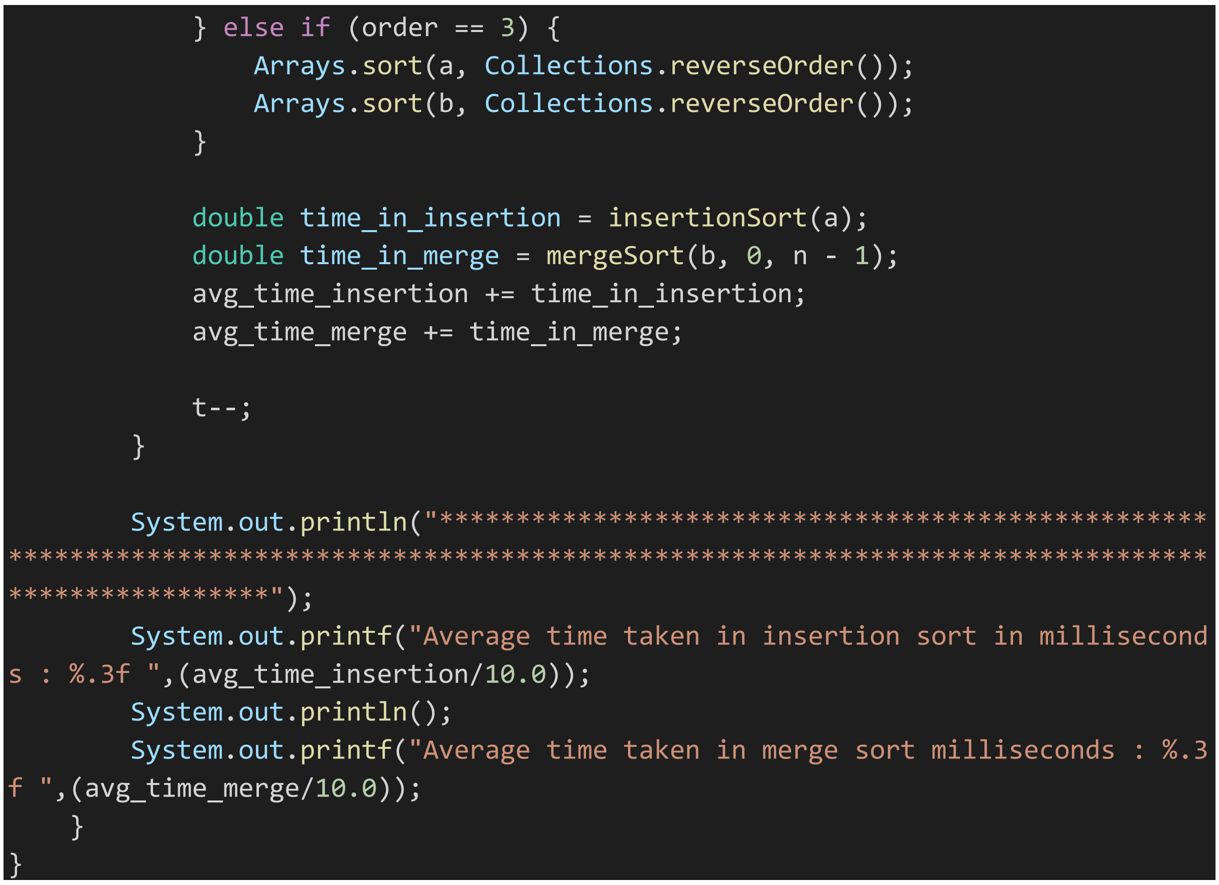
- If the user wants a random sorted to be passed to the sorting methods(insertion sort, merge sort), this array can directly be passed.
- If the user wants an ascending sorted array, this array will be first sorted in ascending order and then it will be passed to the sorting methods.

- If the user wants a descending sorted array, this array will be first sorted in descending order and then it will be passed to the sorting methods.

The program will calculate time taken to sort the array for both insertion and merge sort 10 times and gives the output as average time taken to do so for both the sorting methods.

CODE

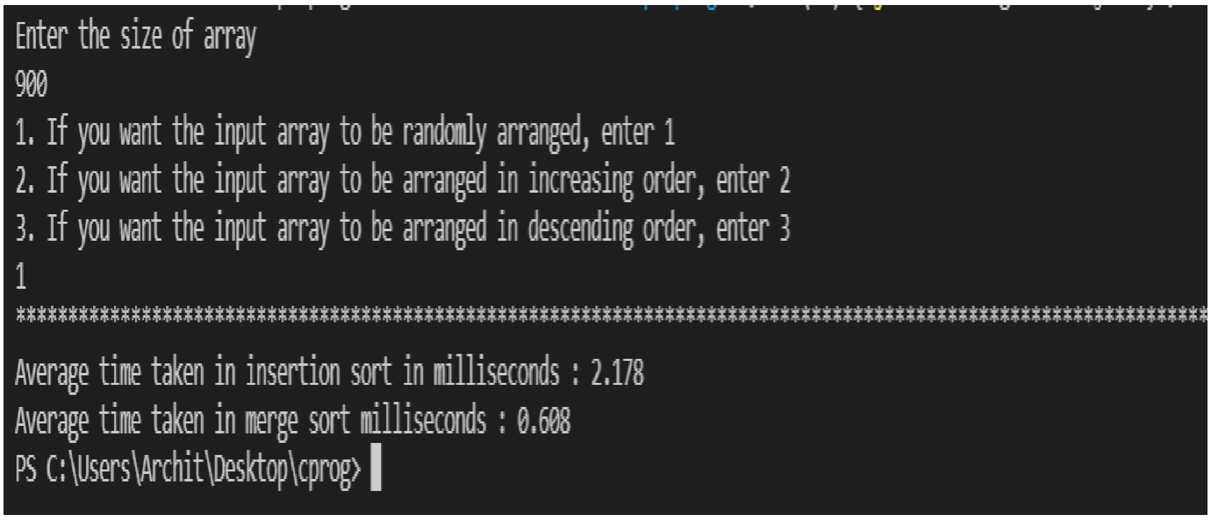
```java
/* This code is written to compare the average time between insertion sort and
 merge sort algorithms
   You will be asked to enter the size of array as input
   You will be asked to decide the order of input array
   an array will be generated randomly of that size and time will be calculate
d for both the algorithms (this will take place 10 times)
   the average of the time for both the methods will be calculated and printed
 */
import java.util.*;
import java.util.Random;
import java.util.Arrays;
import java.util.Collections;

public class SortingMethods{

    public static double insertionSort(Integer []arr){
        double start = System.nanoTime();
        //Insertion Sort Algorithm
        int key;
        int n = arr.length;
        for (int i = 1; i < n; i++){
            key = arr[i];
            int j = i - 1;
            while (j >= 0 && arr[j] > key){
                arr[j + 1] = arr[j];
                j = j - 1;
            }
            arr[j + 1] = key;
        }
        double end = System.nanoTime();
        return (end - start)/1000000.0;
    }
    public static void merge(Integer[] arr, int l, int m, int r){
        // sizes of two subarrays to be merged
        int s1 = m - l + 1;
        int s2 = r - m;

        int[] arr1 = new int [s1];
        int[] arr2 = new int [s2];

        int i, j;
        for (i = 0; i < s1; i++){
            arr1[i] = arr[l + i];
        }
        for (j = 0; j < s2; j++){
            arr2[j] = arr[m + 1 + j];
        }

        i = 0; j = 0;

        int k = l;
        while (i < s1 && j < s2){
            if (arr1[i] <= arr2[j]){
                arr[k] = arr1[i];
                i++;
            }
            else{
                arr[k] = arr2[j];
                j++;
            }
            k++;
        }

        while (i < s1){
            arr[k] = arr1[i];
            i++;
            k++;
        }

        while (j < s2){
            arr[k] = arr2[j];
            j++;
            k++;
        }
    }

    public static double mergeSort(Integer[] arr,int l,int r){
        double start = System.nanoTime();
        if (l < r){
            int m = (l + r)/2;
            mergeSort(arr, l, m);
            mergeSort(arr , m + 1, r);
            merge(arr, l, m, r);
        }
        double end = System.nanoTime();
        return (end - start)/1000000.0;
    }


    public static void main(String []args){
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the size of array");
        int n = sc.nextInt();

        System.out.println("1. If you want the input array to be randomly arra
nged, enter 1");
        System.out.println("2. If you want the input array to be arranged in i
ncreasing order, enter 2");
        System.out.println("3. If you want the input array to be arranged in d
escending order, enter 3");
        int order = sc.nextInt();

        double avg_time_insertion = 0.0;
        double avg_time_merge = 0.0;

        int t = 10;
        while(t > 0) {  //while loop is used to run insertion/merge sort for d
ifferent arrays and compute the average.
            Integer[] a = new Integer[n];
            Integer[] b = new Integer[n]; //a copy of a[]
            Random rand = new Random();

            for (int i = 0; i < n; i++) {
                a[i] = rand.nextInt(9000) + 1000;
                //System.out.print(a[i] + " ");
            }
            for(int i = 0; i < n; i++){
                b[i] = a[i];
            }
            //a[] will be insertion sorted
            //b[] will be merge sorted
            // as random will give a new number everytime that is why
            //a copy of a[] is created to ensure that both the sorting
            //methods gets the same array

            if (order == 2) {
                Arrays.sort(a);
                Arrays.sort(b);
```



```
            } else if (order == 3) {
                Arrays.sort(a, Collections.reverseOrder());
                Arrays.sort(b, Collections.reverseOrder());
            }

            double time_in_insertion = insertionSort(a);
            double time_in_merge = mergeSort(b, 0, n - 1);
            avg_time_insertion += time_in_insertion;
            avg_time_merge += time_in_merge;

            t--;
        }

        System.out.println("**************************************************
******************************************************************************
*****************");
        System.out.printf("Average time taken in insertion sort in millisecond
s : %.3f ",(avg_time_insertion/10.0));
        System.out.println();
        System.out.printf("Average time taken in merge sort milliseconds : %.3
f ",(avg_time_merge/10.0));
    }
}
```

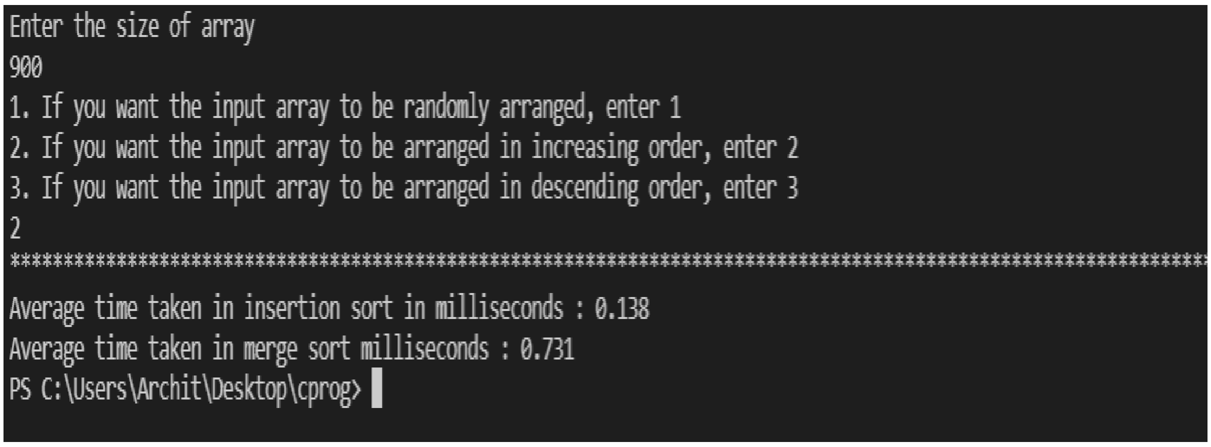
OUTPUT

- ***For random ordered input array***



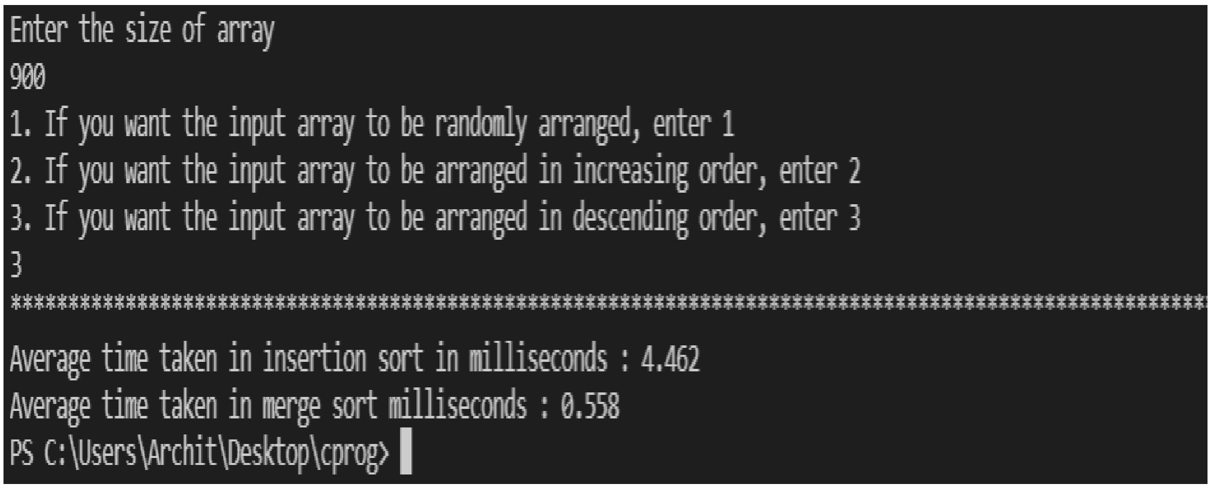
```
Enter the size of array
900
1. If you want the input array to be randomly arranged, enter 1
2. If you want the input array to be arranged in increasing order, enter 2
3. If you want the input array to be arranged in descending order, enter 3
1
************************************************************************************************************************

Average time taken in insertion sort in milliseconds : 2.178
Average time taken in merge sort milliseconds : 0.608
PS C:\Users\Archit\Desktop\cprog>
```

- ***For ascending ordered input array***



```
Enter the size of array
900
1. If you want the input array to be randomly arranged, enter 1
2. If you want the input array to be arranged in increasing order, enter 2
3. If you want the input array to be arranged in descending order, enter 3
2
****************************************************************************************************************

Average time taken in insertion sort in milliseconds : 0.138
Average time taken in merge sort milliseconds : 0.731
PS C:\Users\Archit\Desktop\cprog>
```

- ***For descending ordered input array***



```
Enter the size of array
900
1. If you want the input array to be randomly arranged, enter 1
2. If you want the input array to be arranged in increasing order, enter 2
3. If you want the input array to be arranged in descending order, enter 3
3
****************************************************************************************************************

Average time taken in insertion sort in milliseconds : 4.462
Average time taken in merge sort milliseconds : 0.558
PS C:\Users\Archit\Desktop\cprog>
```

The above data is tabularized below.

S.No.	Sorting Algorithm	Theoretical Time Complexity	Number of elements in the array	Average Time (in ms) for Random Order Input	Average Time (in ms) for Ascending Order Input	Average Time (in ms) for Descending Order Input
1.	Insertion Sort	$O(n^2)$	900	2.178	0.138	4.462
2.	Merge Sort	$O(n \log n)$	900	0.608	0.731	0.558