

**UNIVERSITY INSTITUTE OF ENGINEERING**

**Department of Computer Science & Engineering**

**Subject Name:** Competitive Coding

**Subject Code:** 20CSP-314

**Submitted to: Submitted by:**

Er. Mamta Punia Name: Sahil Kaundal

UID: 21BCS8197

Section: 616

Group: A

**INDEX**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Ex. No** | **List of Experiments** | **Conduct (MM: 12)** | **Viva**  **(MM: 10)** | **Record (MM: 8)** | **Total**  **(MM: 30)** | **Remarks/Signature** |
| 1 | To demonstrate the concept of Array. |  |  |  |  |  |
| 2 | To demonstrate the concept of Stack and Queue. |  |  |  |  |  |
| 3 | **To demonstrate the concept of Linked List.** |  |  |  |  |  |
| 4 | Sorting and Searching: Implement the concept of Searching and Sorting techniques. |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

**Experiment 4.1**

**Student Name:** Sahil Kaundal **UID:** 21BCS8197

**Branch:** BE CSE (Lateral Entry) **Section/Group:** 616/A

**Semester:** 5th **Date of Performance:** 16/09/2022

**Subject Name:** CC Lab **Subject Code:** 20CSP-314

1. **Aim/Overview of the practical:**

**Sorting and Searching: Implement the concept of Searching and Sorting techniques.**

The previous challenges covered [Insertion Sort](https://en.wikipedia.org/wiki/Insertion_sort), which is a simple and intuitive sorting algorithm with a running time of O(n^2). In these next few challenges, we're covering a divide-and-conquer algorithm called [Quicksort](https://en.wikipedia.org/wiki/Quicksort) (also known as Partition Sort). This challenge is a modified version of the algorithm that only addresses partitioning.

<https://www.hackerrank.com/challenges/quicksort1/problem?isFullScreen=true>

1. **Apparatus / Simulator Used:**

* Windows 7 or above
* Google Chrome

1. **Objective:**

* To learn about Searching and sorting in data Structure.
* To learn different approaches, solve fraudulent activity notifications.

**4. Code:**

#include <map>

#include <set>

#include <list>

#include <cmath>

#include <ctime>

#include <deque>

#include <queue>

#include <stack>

#include <bitset>

#include <cstdio>

#include <vector>

#include <cstdlib>

#include <numeric>

#include <sstream>

#include <iostream>

#include <algorithm>

using namespace std;

void swap(vector<int> &arr, int i, int j){

    if(i==j) return;

    int tmp = arr[i];

    arr[i] = arr[j];

    arr[j] = tmp;

}

/\* Head ends here \*/

void partition(vector <int>  ar) {

    int boundary =0, i=1, number\_of\_ele= ar.size(), tmp=0;

    while(i<number\_of\_ele){

        if(ar[i] < ar[0]) {

            tmp = ar[i];

            for(int j = i;j>boundary+1;j--){

                ar[j] = ar[j-1];

            }

            boundary++;

            ar[boundary] = tmp;

        }

        i++;

    }

    int pivot = ar[0];

    for(i=0;i<boundary;i++)

        ar[i] = ar[i+1];

    ar[boundary] = pivot;

    for(i =0; i < number\_of\_ele;i++){

        if(i < number\_of\_ele -1) cout<<ar[i]<<" ";

        else cout<<ar[i]<<endl;

    }

}

/\* Tail starts here \*/

int main() {

   vector <int>  \_ar;

   int \_ar\_size;

cin >> \_ar\_size;

for(int \_ar\_i=0; \_ar\_i<\_ar\_size; \_ar\_i++) {

   int \_ar\_tmp;

   cin >> \_ar\_tmp;

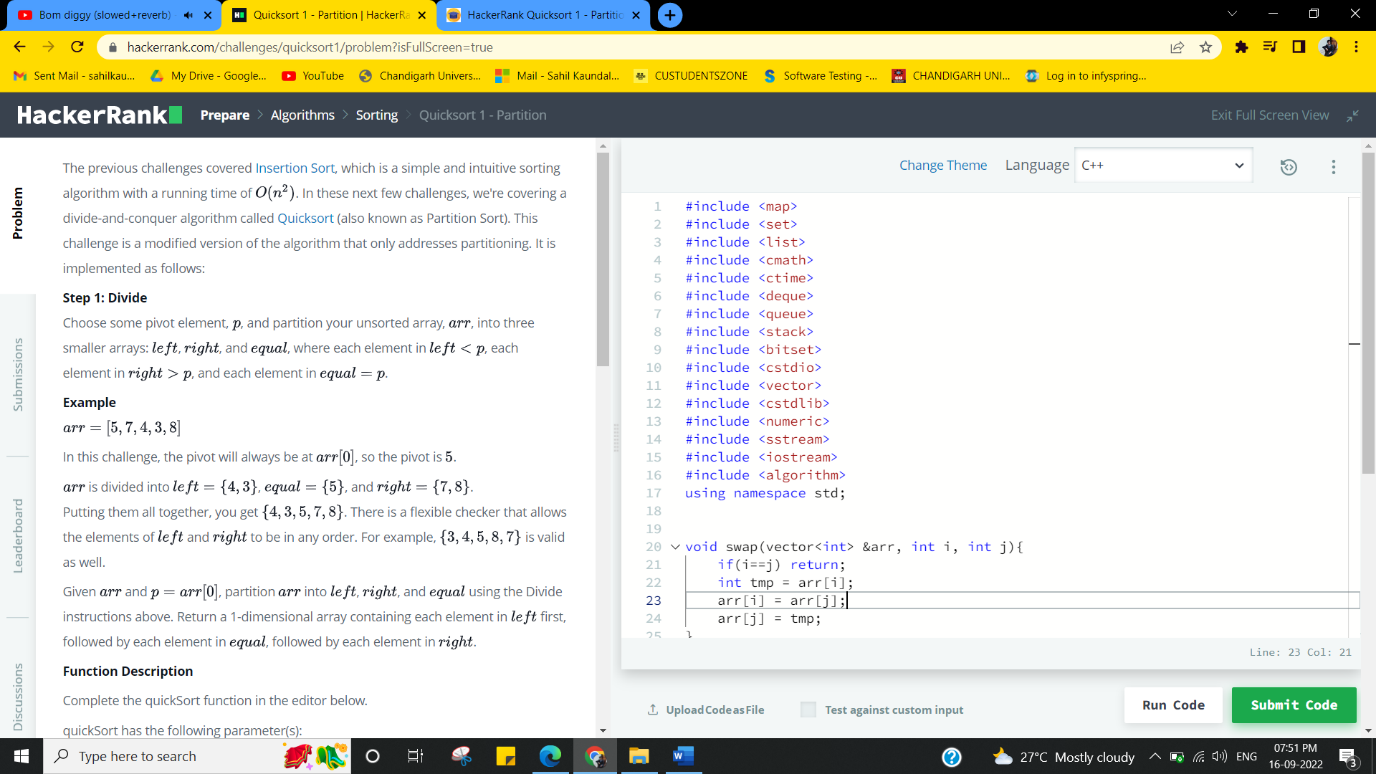
   \_ar.push\_back(\_ar\_tmp);

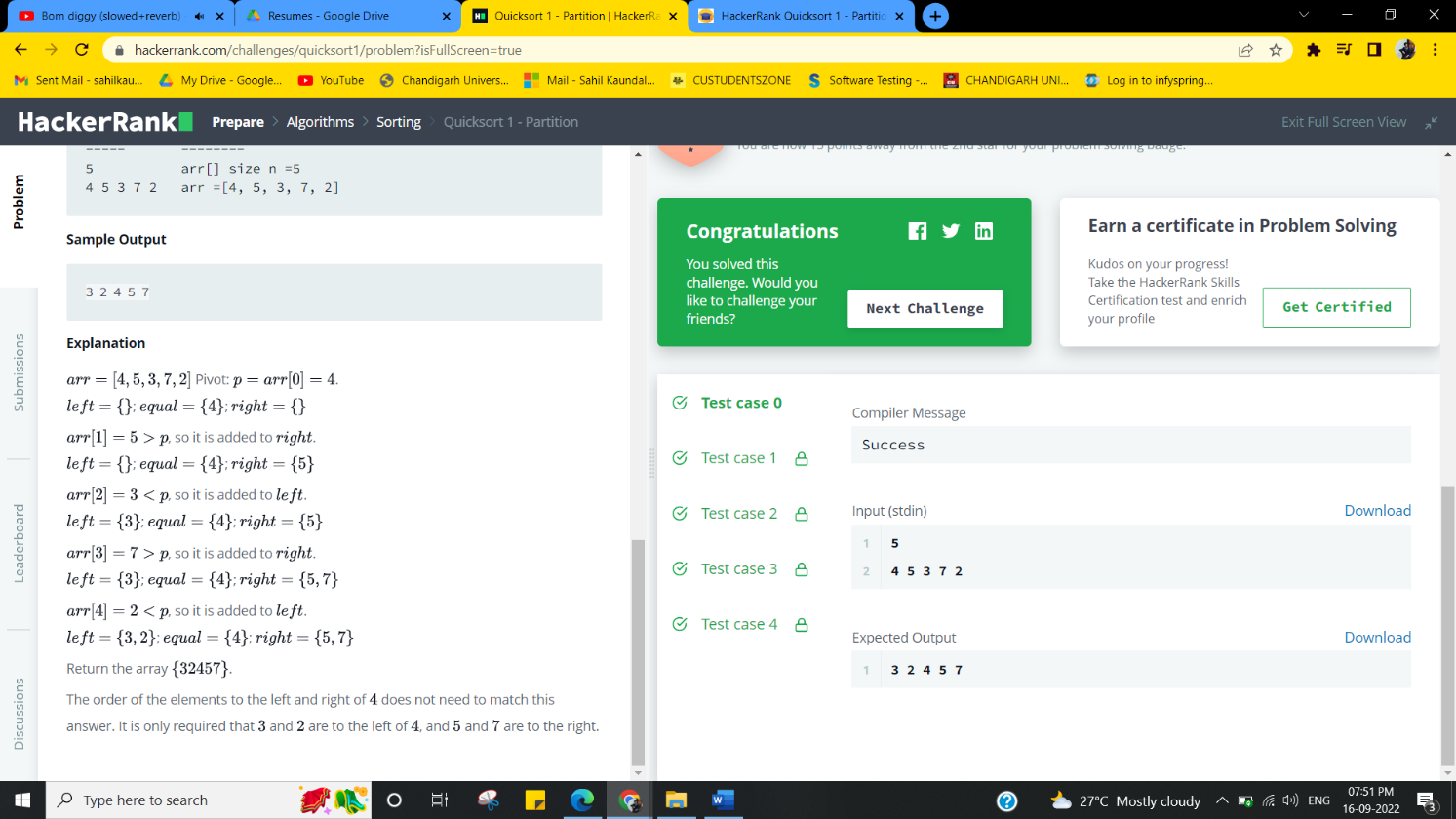
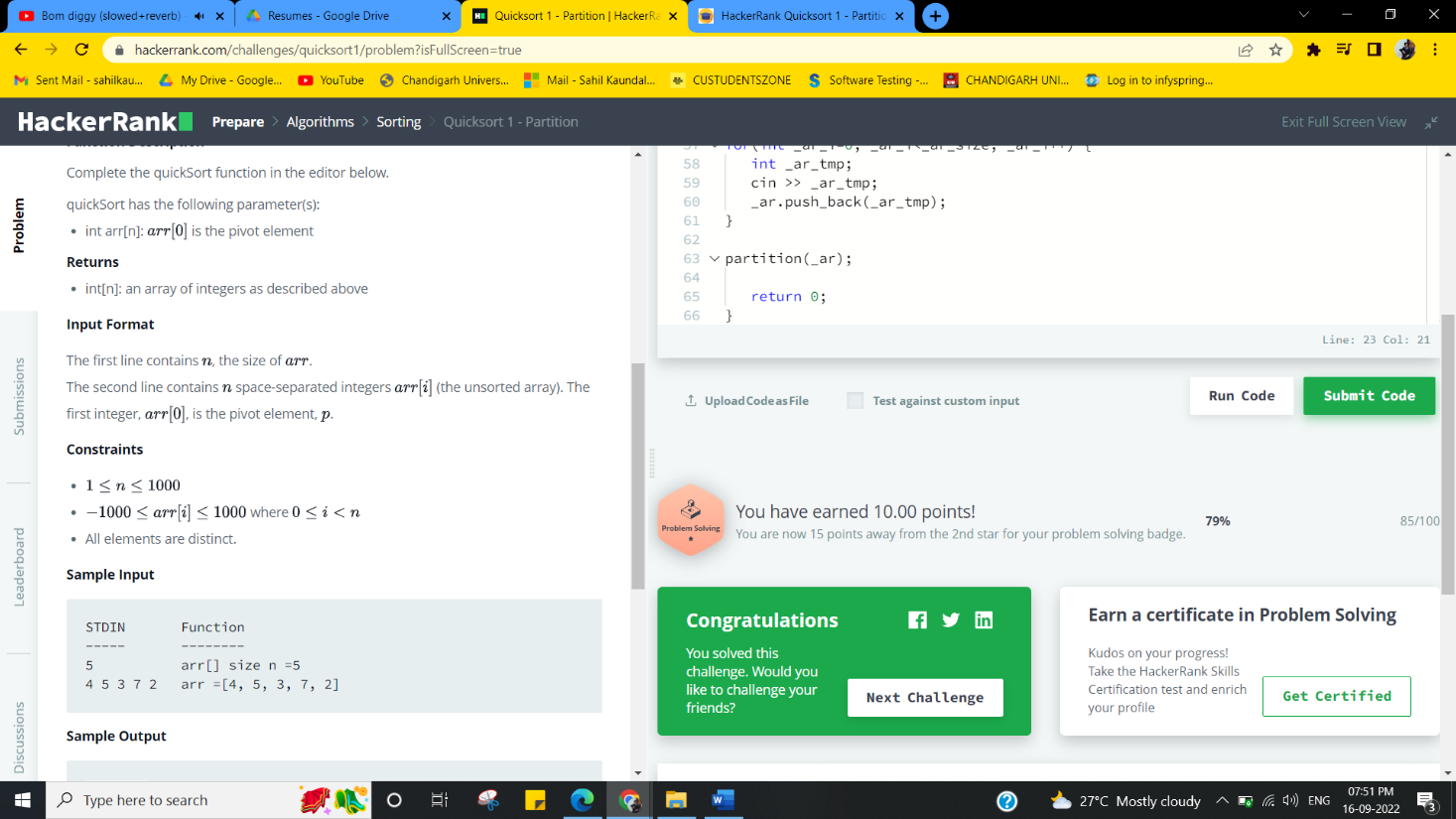
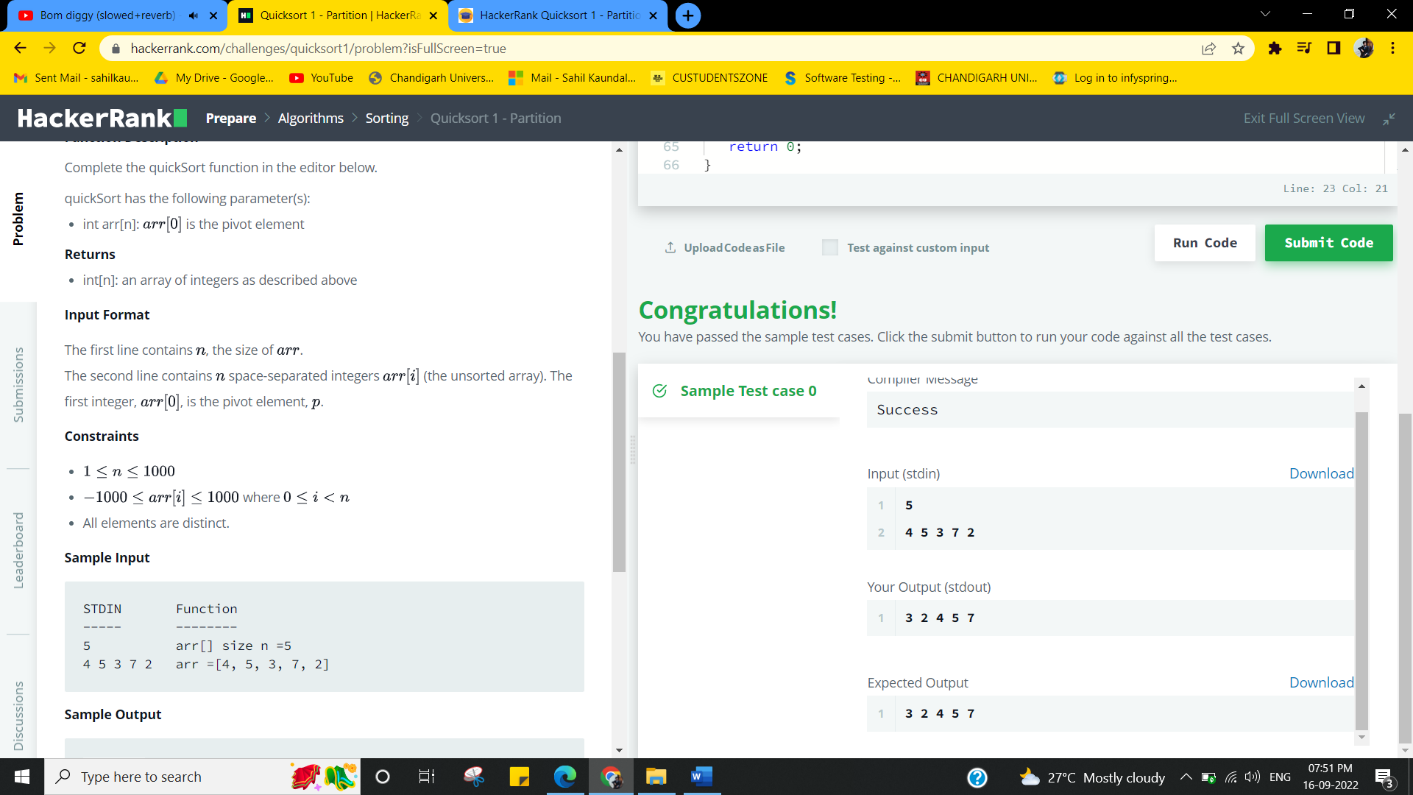
}

partition(\_ar);

   return 0;

}

**5. Result/Output/Writing Summary:**

****

**Experiment 4.2**

1. **Aim/Overview of the practical:**

**Sorting and Searching: Implement the concept of Searching and Sorting techniques.**

Insertion Sort is a simple sorting technique which was covered in previous challenges. Sometimes, arrays may be too large for us to wait around for insertion sort to finish. Is there some other way we can calculate the number of shifts an insertion sort performs when sorting an array?

<https://www.hackerrank.com/challenges/insertion-sort/problem?isFullScreen=true>

1. **Apparatus / Simulator Used:**

* Windows 7 or above
* Google Chrome

1. **Objective:**

* To learn about Searching and sorting in data Structure.
* To learn different approaches solve fraudulent activity notifications.

1. **Code:**

import java.io.\*;

import java.util.\*;

public class Solution {

    private static final int MAXVAL = 10000000;

    private static int[] array = new int[MAXVAL+1];

    public static void main(String[] args) {

        /\* Enter your code here. Read input from STDIN. Print output to STDOUT. Your class should be named Solution. \*/

        Scanner sc = new Scanner(System.in);

        int testCaseCount = sc.nextInt();

        for (int i = 0; i < testCaseCount; i++) {

            int size = sc.nextInt();

            long sum = 0;

            Arrays.fill(array, 0);

            for (int j = 0; j < size; j++) {

                 sum += assign(sc.nextInt(), array, j);

            }

            System.out.println(sum);

        }

    }

    private static int assign(int x, int[] prefixSums, int current) {

        int n = read(prefixSums, x);

        update(prefixSums, x);

        return current-n;

    }

    private static int read(int[] prefixSums, int x) {

        int nrt=0;

        while(x>0) {

            nrt += prefixSums[x];

            x -= (x&(-x));

        }

        return nrt;

    }

    private static void update(int[] prefixSums, int x) {

        while(x <= MAXVAL) {

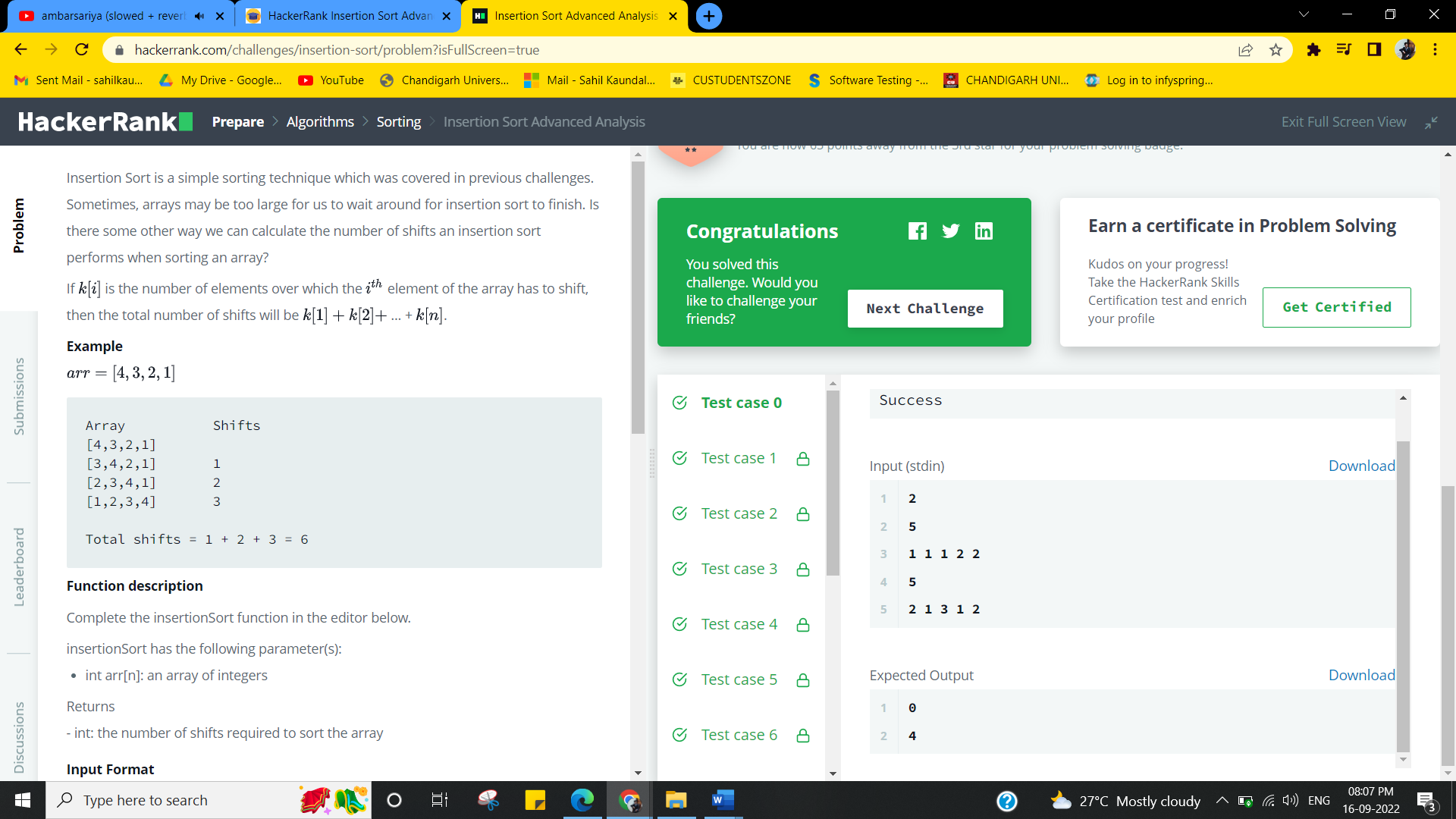
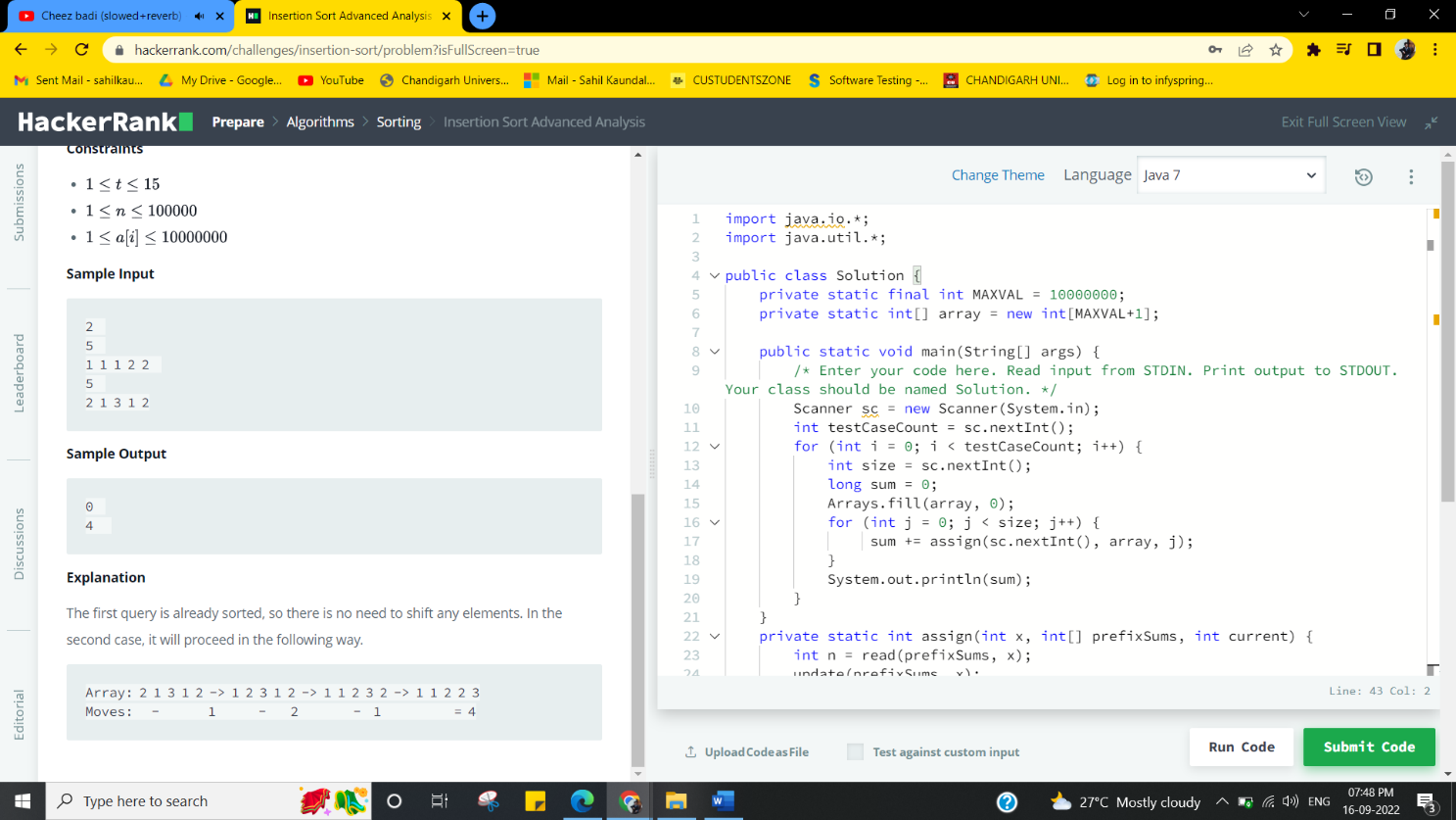
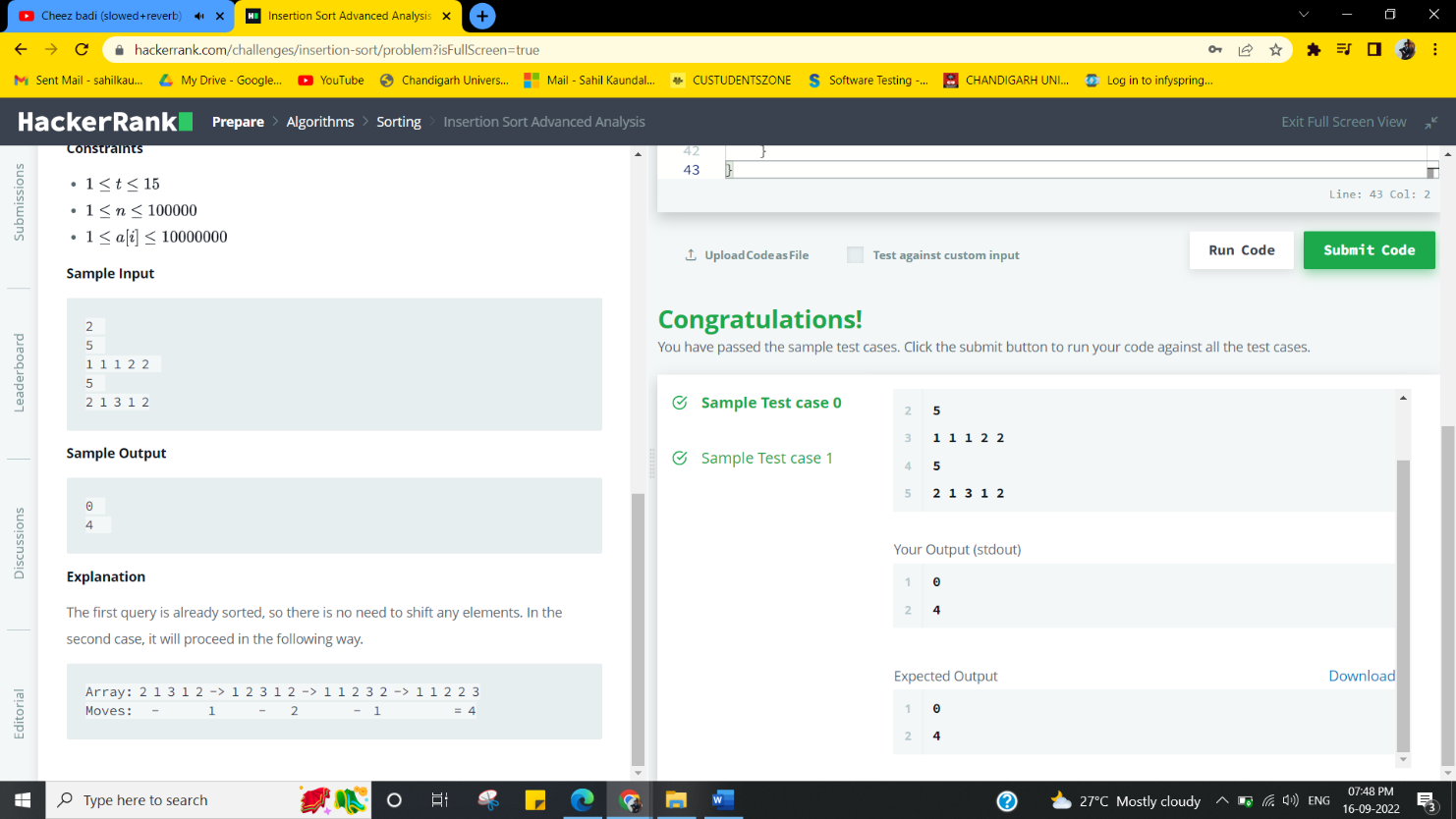
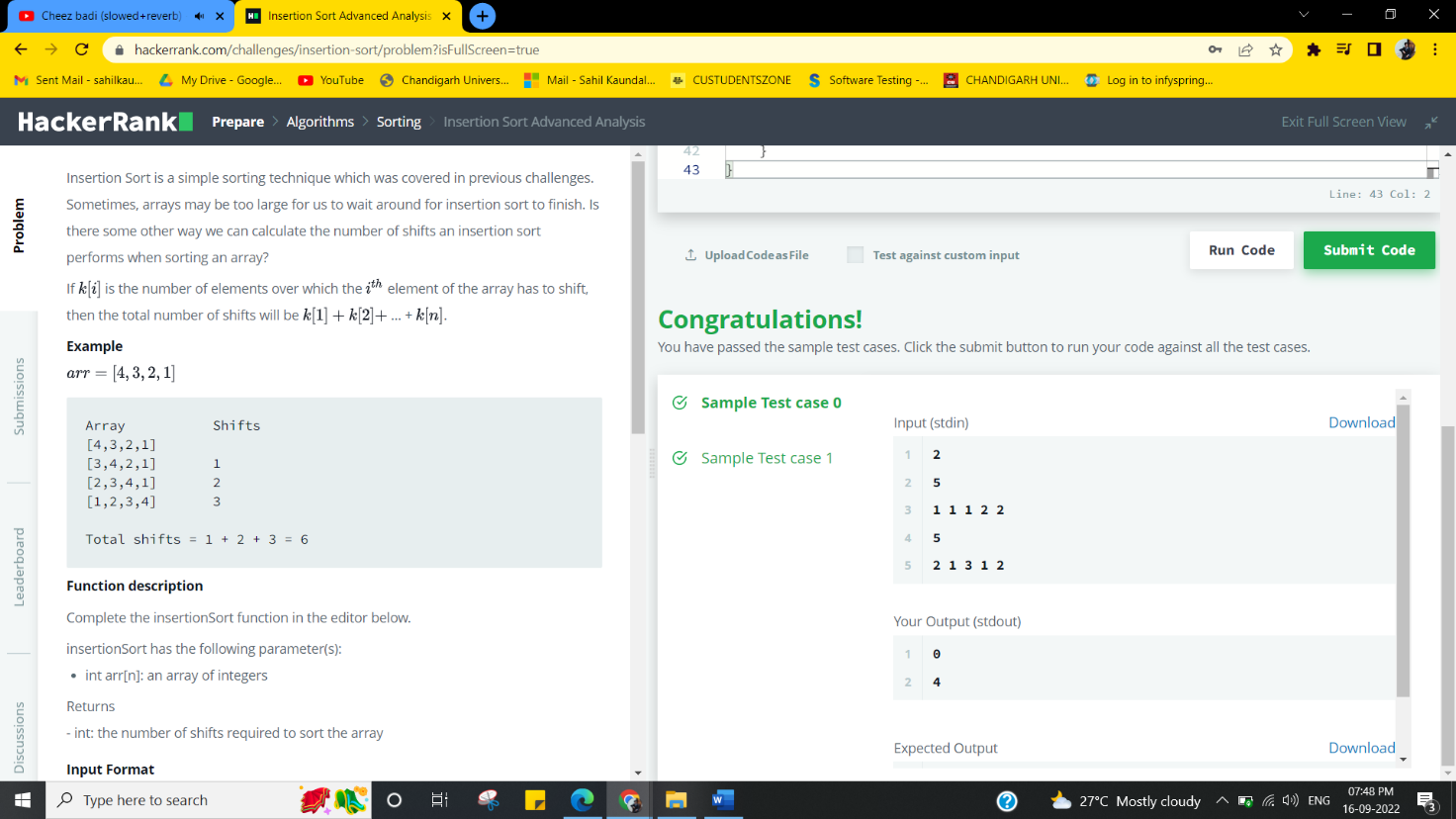
            prefixSums[x]++;

            x += (x&(-x));

        }

    }

}

1. **Result/Output/Writing Summary:**

**Learning outcomes (What I have learnt):**

# Learnt different approaches, solve fraudulent activity notifications.

# Implemented the concept of Searching and Sorting techniques.

**Evaluation Grid (To be created as per the SOP and Assessment guidelines by the faculty):**

|  |  |  |  |
| --- | --- | --- | --- |
| Sr. No. | Parameters | Marks Obtained | Maximum Marks |
| 1. |  |  |  |
| 2. |  |  |  |
| 3. |  |  |  |
|  |  |  |  |