## CHANDIGARH UNIVERSITY

## UNIVERSITY INSTITUTE OF NGINEERING

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**



|  |  |
| --- | --- |
| **Submitted By: Submitted To:**  Vivek Kumar(21BCS8129) Mamta Punia(E12337) | |
| **Subject Name** | Competitive Coding - I |
| **Subject Code** | 20CSP-314 |
| **Branch** | Computer Science and Engineering |
| **Semester** | 5th |

**Experiment No. - 4**

**Student Name: Vivek Kumar UID: 21BCS8129**

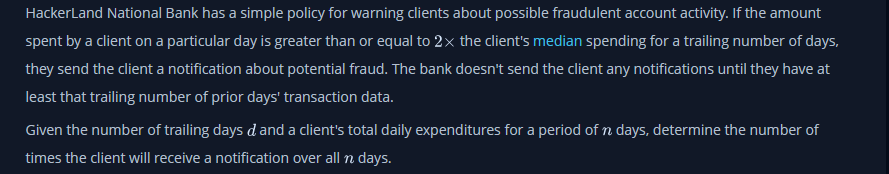
**Branch: BE-CSE(LEET) Section/Group: WM-20BCS-616/A**

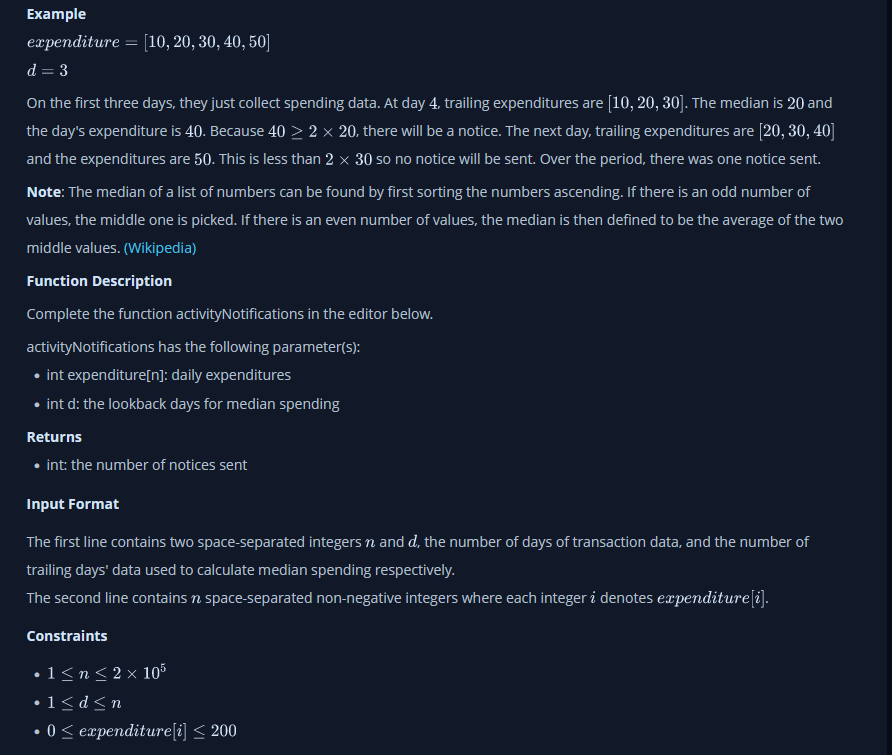
**Semester: 5th Date of Performance: 02/09/2022**

**Subject Name: Competitive coding - I Subject Code: 20CSP-314**

# **Fraudulent Activity Notifications:**

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

**2. Task to be done/ Which logistics used:**





**3. Hardware and Software Requirements (For programming-based labs):**

* Laptop or Desktop
* Hacker-Rank Account

**4. Steps for experiment/practical/Code:**

import java.io.\*;

import java.util.\*;

public class Solution {

    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);

        int n = input.nextInt();

        int d = input.nextInt();

        int notifications = 0;

        Queue < Integer > queue = new LinkedList < > ();

        int[] pastActivity = new int[201];

        for (int i = 0; i < d; i++) {

            int transaction = input.nextInt();

            queue.offer(transaction);

            pastActivity[transaction] = pastActivity[transaction] + 1;

        }

        for (int i = 0; i < n - d; i++) {

            int newTransaction = input.nextInt();

            if (newTransaction >= (2 \* median(pastActivity, d))) notifications++;

            int oldestTransaction = queue.poll();

            pastActivity[oldestTransaction] = pastActivity[oldestTransaction] - 1;

            queue.offer(newTransaction);

            pastActivity[newTransaction] = pastActivity[newTransaction] + 1;

        }

        System.out.println(notifications);

    }

    static double median(int[] array, int elements) {

        int index = 0;

        if (elements % 2 == 0) {

            int counter = (elements / 2);

            while (counter > 0) {

                counter -= array[index];

                index++;

            }

            index--;

            if (counter <= -1) return index;

            else {

                int firstIndex = index;

                int secondIndex = index + 1;

                while (array[secondIndex] == 0) {

                    secondIndex++;

                }

                return (double)(firstIndex + secondIndex) / 2.0;

            }

        } else {

            int counter = (elements / 2);

            while (counter >= 0) {

                counter -= array[index];

                index++;

            }

            return (double) index - 1;

        }

    }

    static void printArray(int[] array) {

        System.out.println("Array");

        for (int i = 0; i < array.length; i++) {

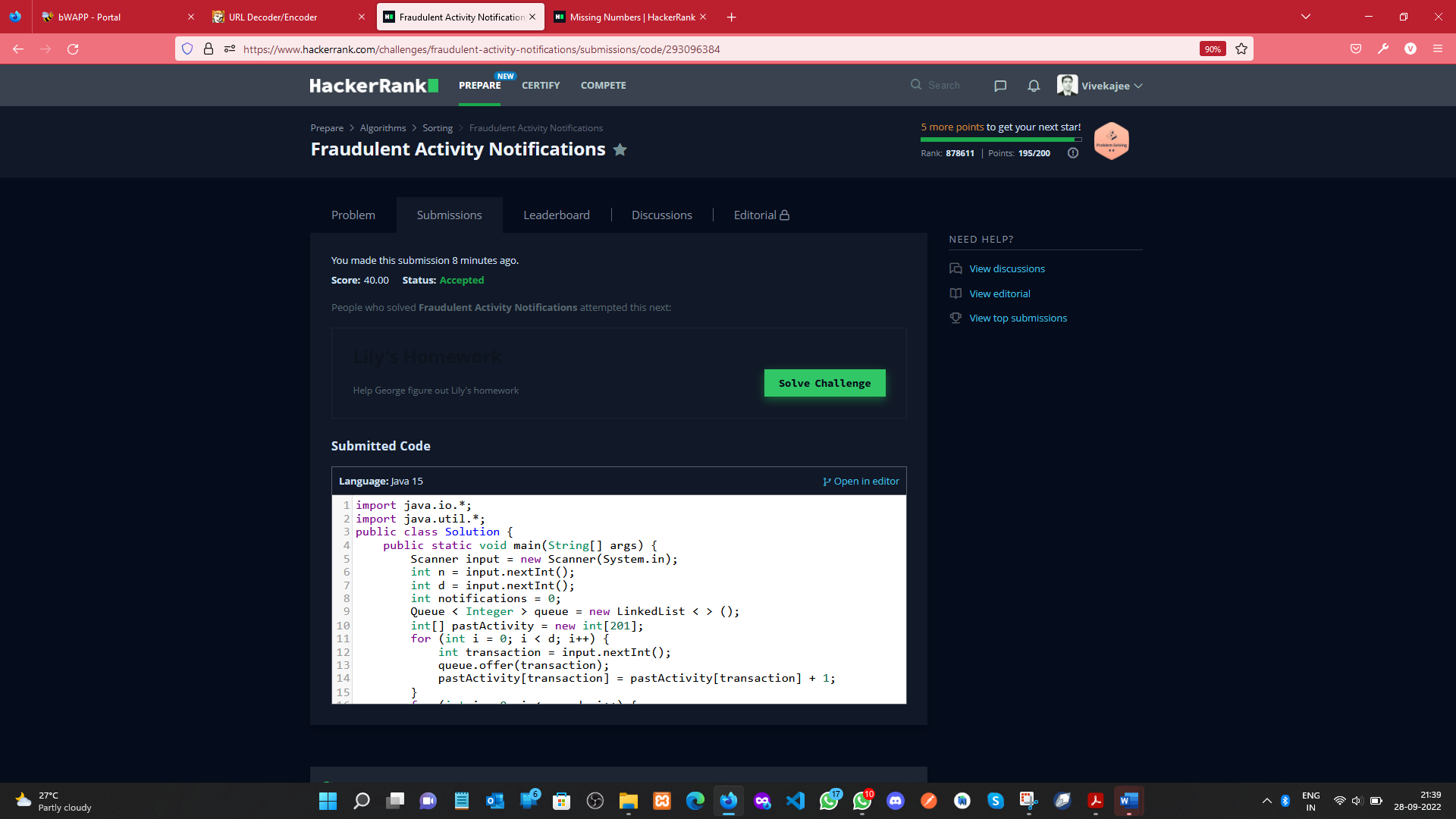
            if (array[i] > 0) System.out.println(i + " : " + array[i]);

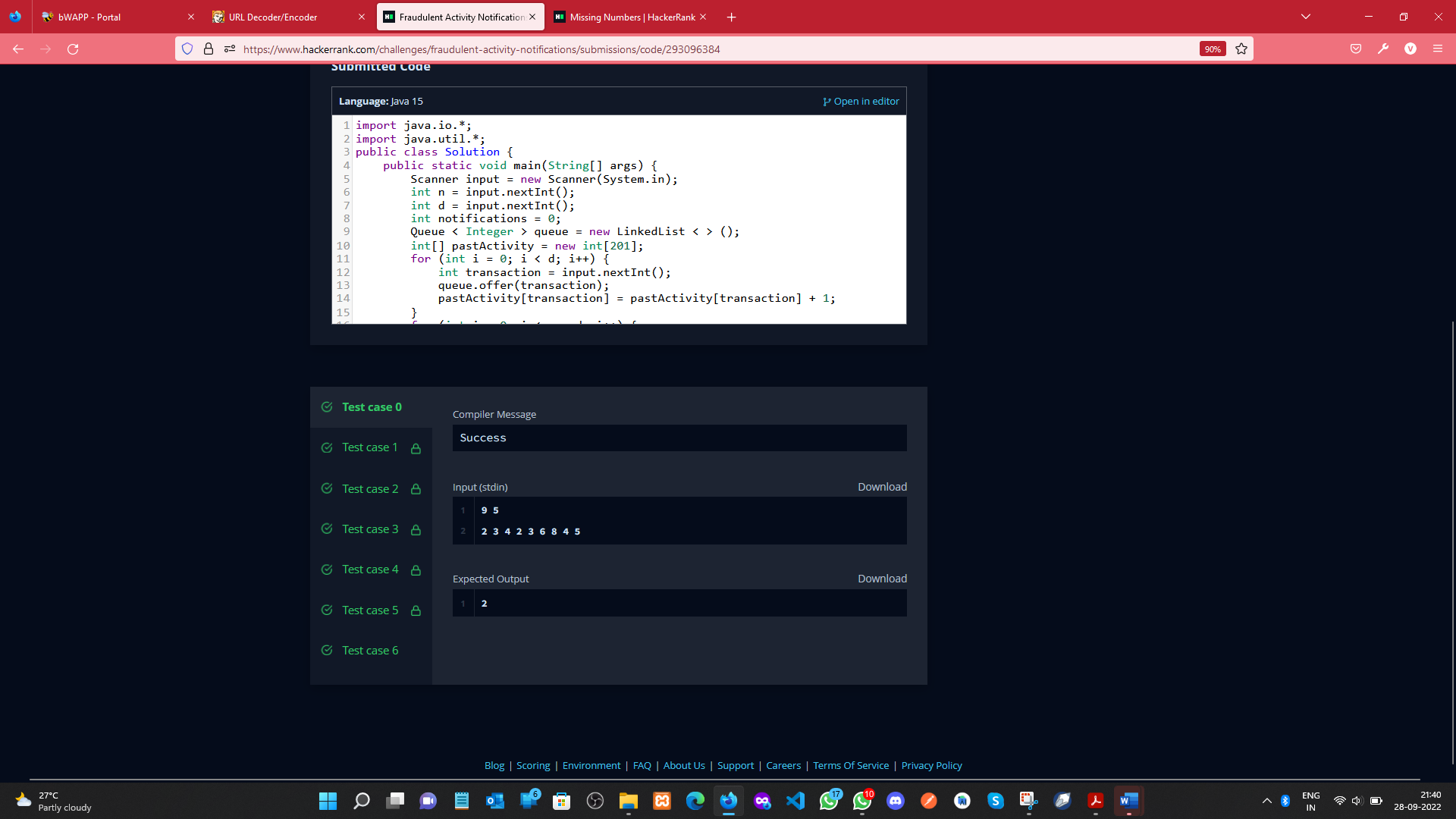
        }

    }

}

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





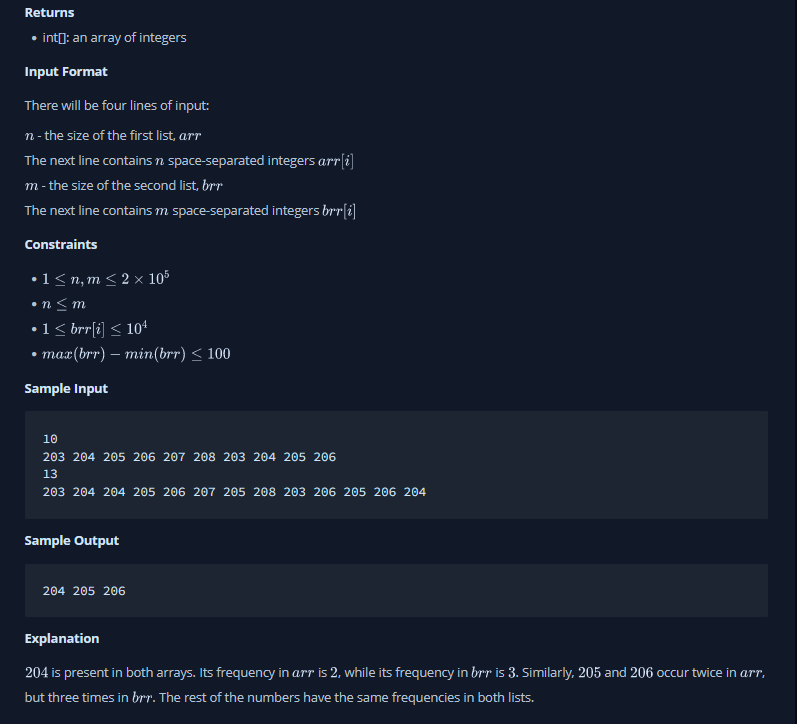
**Missing Numbers:**

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

Given two arrays of integers, find which elements in the second array are missing from the first array.

**2. Task to be done/ Which logistics used:**





**3. Hardware and Software Requirements (For programming-based labs):**

* Laptop or Desktop
* Hacker-Rank Account

**4. Steps for experiment/practical/Code:**

public static List<Integer> missingNumbers(List<Integer> arr, List<Integer> brr) {

// Write your code here

List<Integer> result= new ArrayList<>();

Map<Integer,Integer> fbrr=new HashMap<>();

for(Integer val:brr){

fbrr.put(val,fbrr.getOrDefault(val, 0)+1);

}

for(Integer val:arr){

if(fbrr.containsKey(val)){

fbrr.put(val,fbrr.getOrDefault(val, 0)-1);

}

}

for(Map.Entry<Integer,Integer> entry:fbrr.entrySet()){

if(entry.getValue()>0){

result.add(entry.getKey());

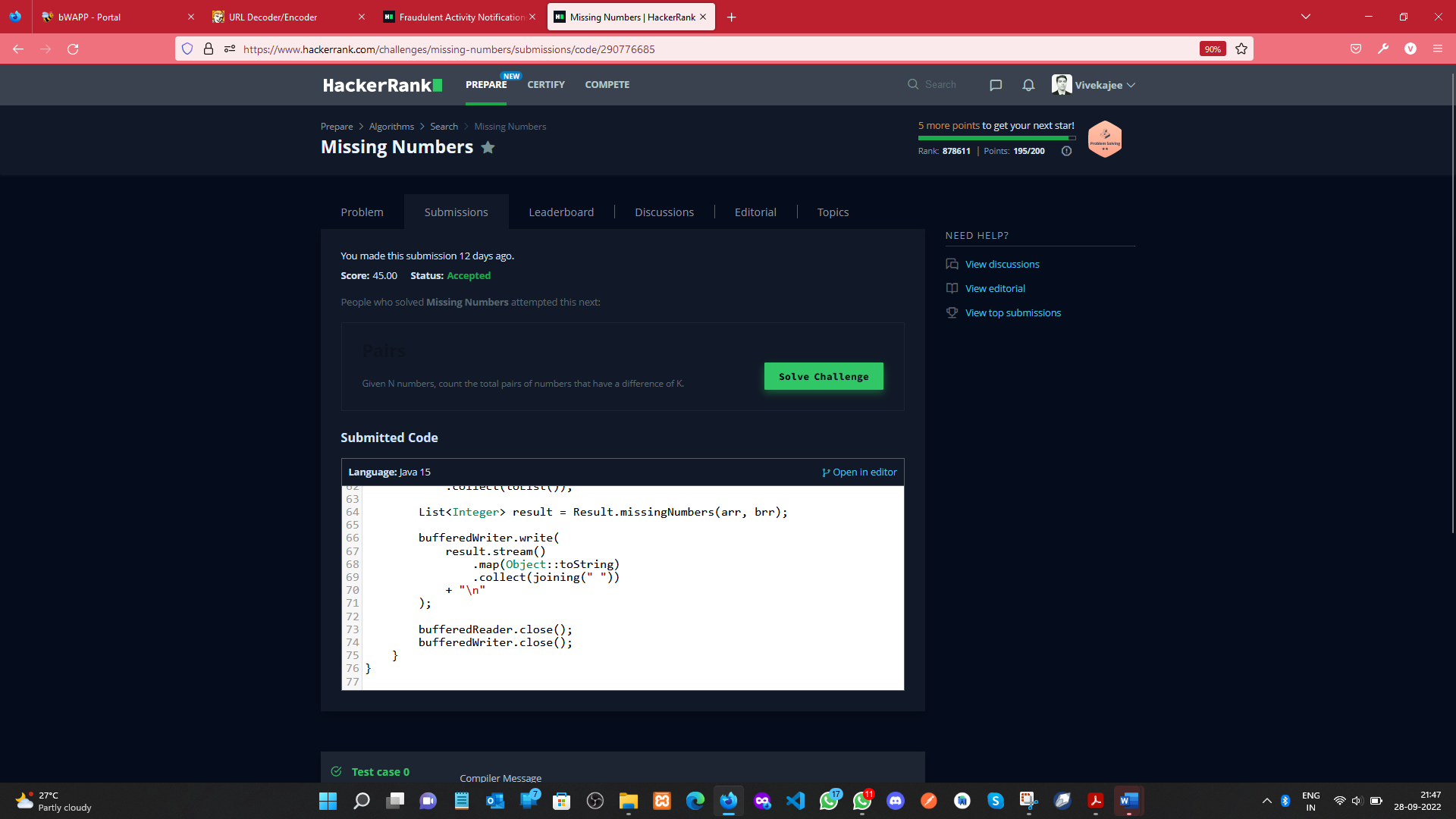
}

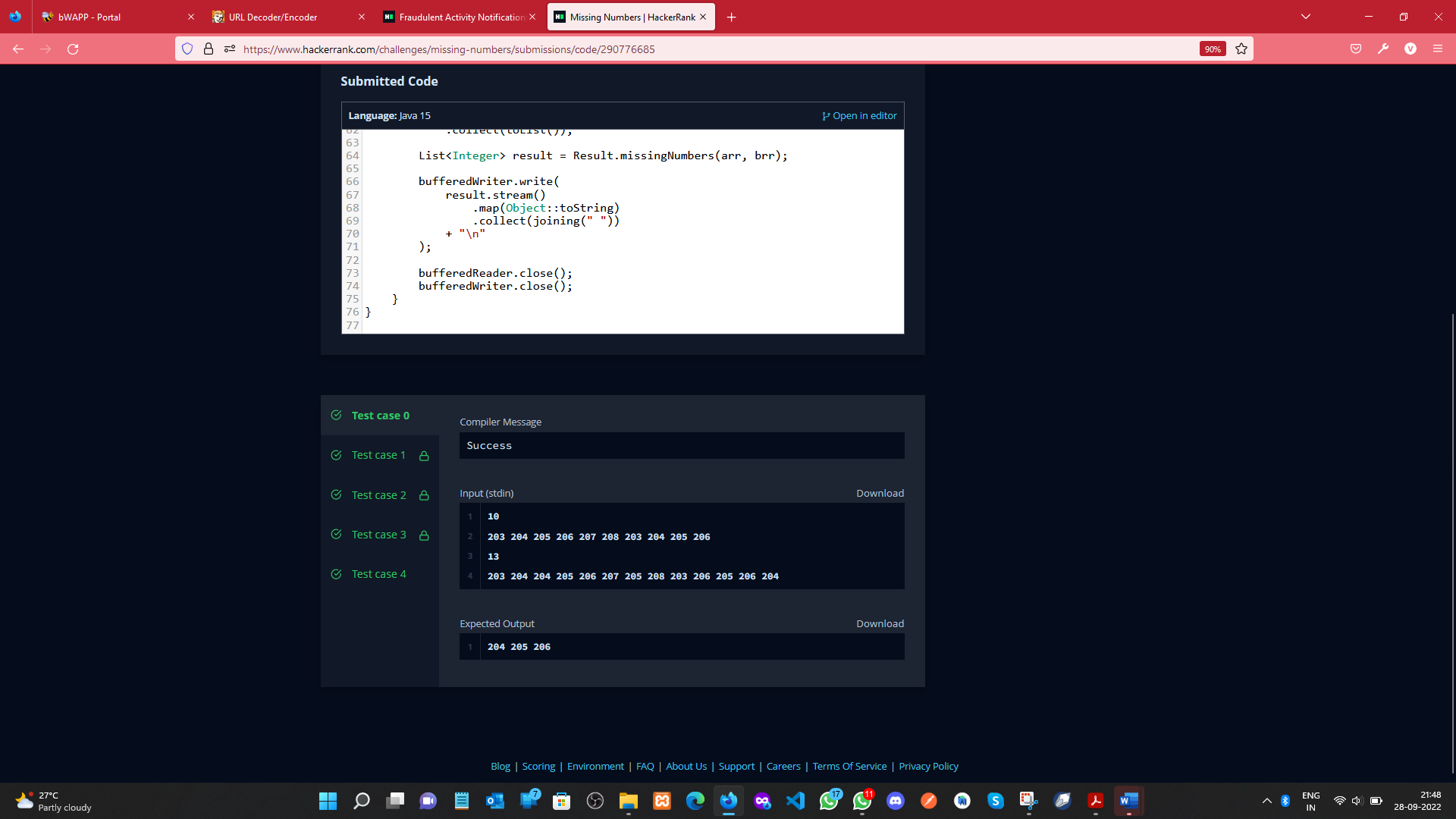
}

return result;

}

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





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

a. Learnt about Vectors.

1. b. Learnt about searching and sorting techniques.
2. c. Got an overview of the type of questions on hacker-rank.
3. d. Get to know about crucial test cases.

**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. |  |  |  |
|  |  |  |  |