CHANDIGARH UNIVERSITY

UNIVERSITY INSTITUTE OF NGINEERING

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Submitted By: Vivek Kumar(21BCS8129)	**Submitted To:** 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:

HackerLand National Bank has a simple policy for warning clients about possible fraudulent account activity. If the amount spent by a client on a particular day is greater than or equal to $2\times$ the client's median spending for a trailing number of days, they send the client a notification about potential fraud. The bank doesn't send the client any notifications until they have at least that trailing number of prior days' transaction data.

Given the number of trailing days d and a client's total daily expenditures for a period of n days, determine the number of times the client will receive a notification over all n days.

2. Task to be done/ Which logistics used:

Example

$expenditure = [10, 20, 30, 40, 50]$

$d = 3$

On the first three days, they just collect spending data. At day 4, trailing expenditures are $[10, 20, 30]$. The median is 20 and the day's expenditure is 40. Because $40 \geq 2 \times 20$, there will be a notice. The next day, trailing expenditures are $[20, 30, 40]$ and the expenditures are 50. This is less than 2×30 so no notice will be sent. Over the period, there was one notice sent.

Note: The median of a list of numbers can be found by first sorting the numbers ascending. If there is an odd number of values, the middle one is picked. If there is an even number of values, the median is then defined to be the average of the two middle values. (Wikipedia)

Function Description

Complete the function activityNotifications in the editor below.

activityNotifications has the following parameter(s):

- int expenditure[n]: daily expenditures
- int d: the lookback days for median spending

Returns

- int: the number of notices sent

Input Format

The first line contains two space-separated integers n and d, the number of days of transaction data, and the number of trailing days' data used to calculate median spending respectively.

The second line contains n space-separated non-negative integers where each integer i denotes $expenditure[i]$.

Constraints

- $1 \leq n \leq 2 \times 10^5$
- $1 \leq d \leq n$
- $0 \leq expenditure[i] \leq 200$

Output Format

Sample Input 0

```
STDIN                Function
-----                --------
9 5                  expenditure[] size n =9, d = 5
2 3 4 2 3 6 8 4 5    expenditure = [2, 3, 4, 2, 3, 6, 8, 4, 5]
```

Sample Output 0

```
2
```

Explanation 0

Determine the total number of $notifications$ the client receives over a period of $n = 9$ days. For the first five days, the customer receives no notifications because the bank has insufficient transaction data: $notifications = 0$.

On the sixth day, the bank has $d = 5$ days of prior transaction data, $\{2, 3, 4, 2, 3\}$, and $median = 3$ dollars. The client spends 6 dollars, which triggers a notification because $6 \geq 2 \times median$: $notifications = 0 + 1 = 1$.

On the seventh day, the bank has $d = 5$ days of prior transaction data, $\{3, 4, 2, 3, 6\}$, and $median = 3$ dollars. The client spends 8 dollars, which triggers a notification because $8 \geq 2 \times median$: $notifications = 1 + 1 = 2$.

On the eighth day, the bank has $d = 5$ days of prior transaction data, $\{4, 2, 3, 6, 8\}$, and $median = 4$ dollars. The client spends 4 dollars, which does not trigger a notification because $4 < 2 \times median$: $notifications = 2$.

On the ninth day, the bank has $d = 5$ days of prior transaction data, $\{2, 3, 6, 8, 4\}$, and a transaction median of 4 dollars. The client spends 5 dollars, which does not trigger a notification because $5 < 2 \times median$: $notifications = 2$.

Sample Input 1

```
5 4
1 2 3 4 4
```

Sample Output 1

```
0
```

There are 4 days of data required so the first day a notice might go out is day 5. Our trailing expenditures are $[1, 2, 3, 4]$ with a median of 2.5 The client spends 4 which is less than 2×2.5 so no notification is sent.

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

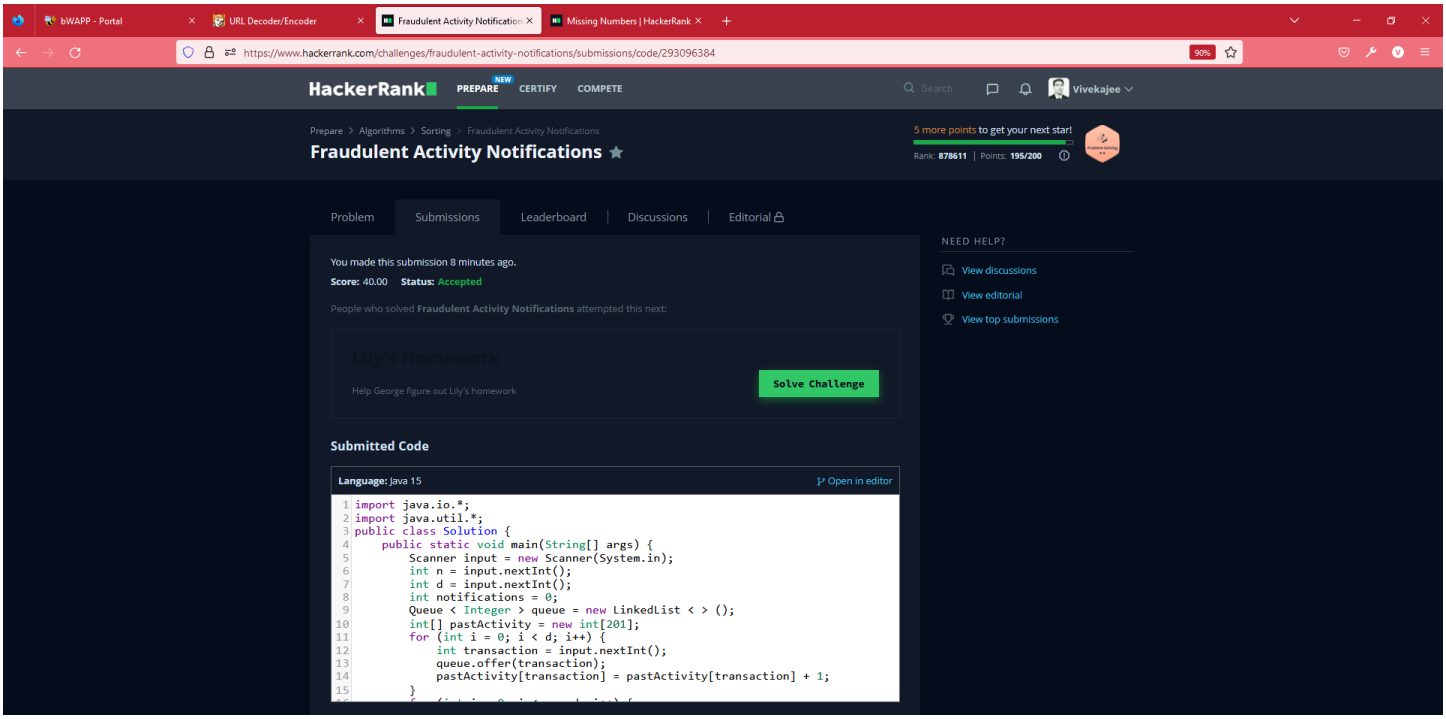
- Laptop or Desktop
- Hacker-Rank Account

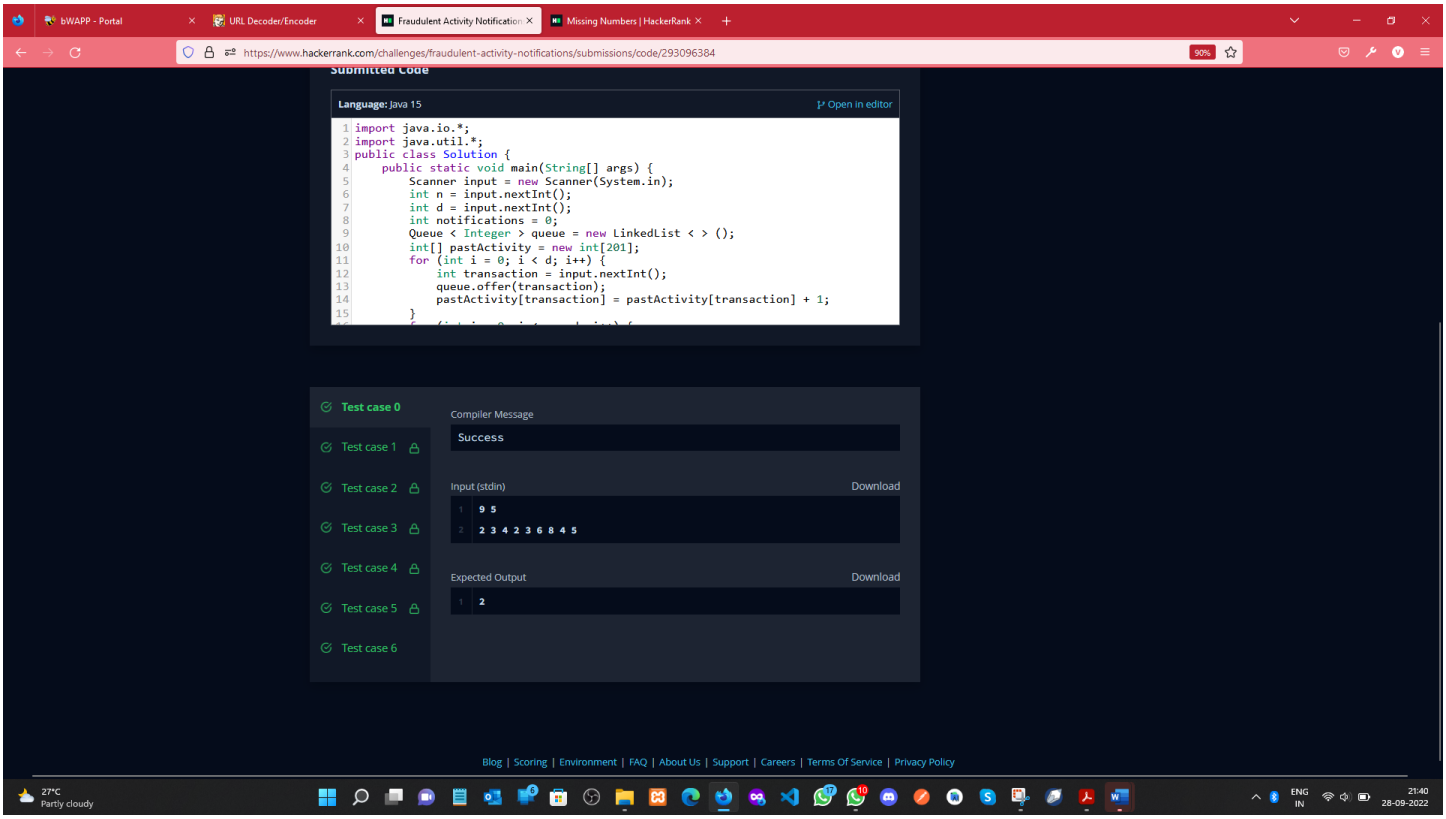
4. Steps for experiment/practical/Code:

```
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:

Example

$arr = [7, 2, 5, 3, 5, 3]$

$brr = [7, 2, 5, 4, 6, 3, 5, 3]$

The brr array is the orginal list. The numbers missing are $[4, 6]$.

Notes

- If a number occurs multiple times in the lists, you must ensure that the frequency of that number in both lists is the same. If that is not the case, then it is also a missing number.
- Return the missing numbers sorted ascending.
- Only include a missing number once, even if it is missing multiple times.
- The difference between the maximum and minimum numbers in the original list is less than or equal to 100.

Function Description

Complete the missingNumbers function in the editor below. It should return a sorted array of missing numbers.

missingNumbers has the following parameter(s):

- int arr[n]: the array with missing numbers
- int brr[m]: the original array of numbers

Returns

- int[]: an array of integers

Input Format

There will be four lines of input:

n - the size of the first list, arr

The next line contains n space-separated integers $arr[i]$

m - the size of the second list, brr

The next line contains m space-separated integers $brr[i]$

Constraints

- $1 \leq n, m \leq 2 \times 10^5$
- $n \leq m$
- $1 \leq brr[i] \leq 10^4$
- $max(brr) - min(brr) \leq 100$

Sample Input

```
10
203 204 205 206 207 208 203 204 205 206
13
203 204 204 205 206 207 205 208 203 206 205 206 204
```

Sample Output

```
204 205 206
```

Explanation

204 is present in both arrays. Its frequency in arr is 2, while its frequency in brr is 3. Similarly, 205 and 206 occur twice in arr, but three times in brr. The rest of the numbers have the same frequencies in both lists.

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

- Laptop or Desktop
- Hacker-Rank Account

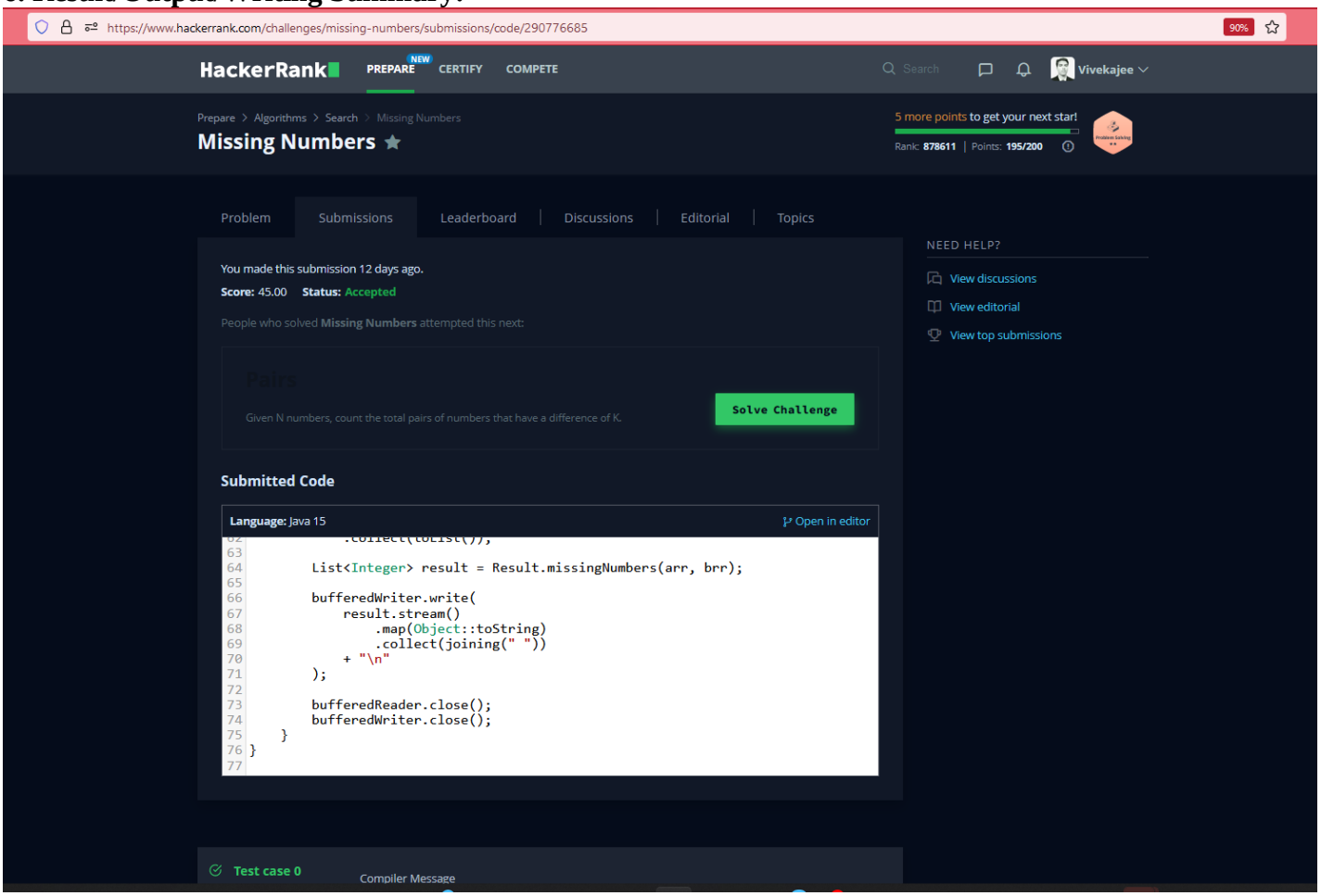
4. Steps for experiment/practical/Code:

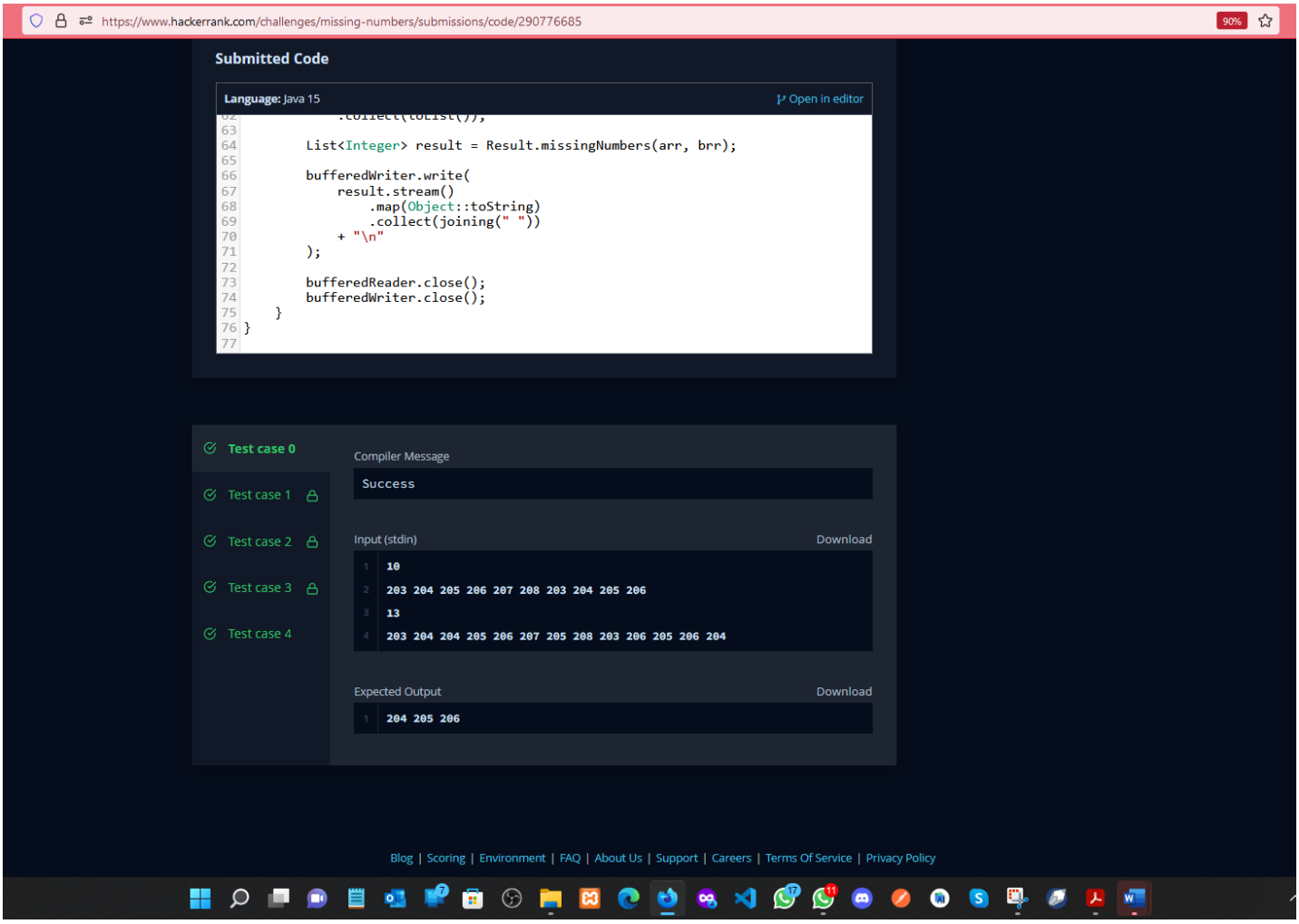
```
public static List<Integer> missingNumbers(List<Integer> arr, List<Integer> brr) {
   // Write your code here
      List<Integer> result= new ArrayList<>();
      Map<Integer,Integer> fbrr=new HashMap<>();
```

```
    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.

b. Learnt about searching and sorting techniques.

c. Got an overview of the type of questions on hacker-rank.

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.			