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. - 2

Student Name: Vivek Kumar **UID: 21BCS8129**
Branch: BE-CSE(LEET) **Section/Group: WM-20BCS-616/A**
Semester: 5th **Date of Performance: 12/08/2022**
Subject Name: Competitive coding - I **Subject Code: 20CSP-314**

Game of Two Stack:

1. Aim/Overview of the practical:

Alexa has two stacks of non-negative integers, stack a[n] and stack b[m] where index 0 denotes the top of the stack. Alexa challenges Nick to play the following game:

- In each move, Nick can remove one integer from the top of either stack a or stack b.
- Nick keeps a running sum of the integers he removes from the two stacks.
- Nick is disqualified from the game if, at any point, his running sum becomes greater than some integer max-Sum given at the beginning of the game.
- Nick's *final score* is the total number of integers he has removed from the two stacks.

Given a, b and max-Sum for g games, find the maximum possible score Nick can achieve.

2. Task to be done/ Which logistics used:

Example

$a = [1, 2, 3, 4, 5]$

$b = [6, 7, 8, 9]$

The maximum number of values Nick can remove is 4. There are two sets of choices with this result.

1. Remove $1, 2, 3, 4$ from a with a sum of 10.
2. Remove $1, 2, 3$ from a and 6 from b with a sum of 12.

Function Description

Complete the twoStacks function in the editor below.

twoStacks has the following parameters: - int maxSum: the maximum allowed sum

- int a[n]: the first stack

- int b[m]: the second stack

Returns

- int: the maximum number of selections Nick can make

Input Format

The first line contains an integer, g (the number of games). The $3 \cdot g$ subsequent lines describe each game in the following format:

1. The first line contains three space-separated integers describing the respective values of n (the number of integers in stack a), m (the number of integers in stack b), and $maxSum$ (the number that the sum of the integers removed from the two stacks cannot exceed).
2. The second line contains n space-separated integers, the respective values of $a[i]$.
3. The third line contains m space-separated integers, the respective values of $b[i]$.

Constraints

- $1 \leq g \leq 50$
- $1 \leq n, m \leq 10^5$
- $0 \leq a[i], b[i] \leq 10^6$
- $1 \leq maxSum \leq 10^9$

Subtasks

- $1 \leq n, m, \leq 100$ for 50% of the maximum score.

Sample Input 0

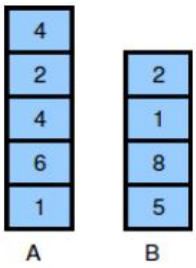
```
1
5 4 10
4 2 4 6 1
2 1 8 5
```

Sample Output 0

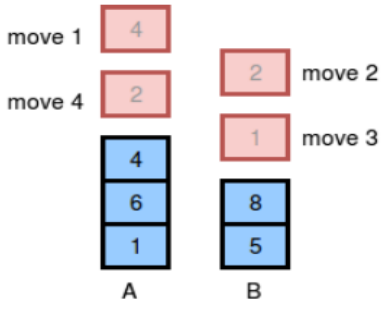
```
4
```

Explanation 0

The two stacks initially look like this:



The image below depicts the integers Nick should choose to remove from the stacks. We print 4 as our answer, because that is the maximum number of integers that can be removed from the two stacks without the sum exceeding $x = 10$.



(There can be multiple ways to remove the integers from the stack, the image shows just one of them.)

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

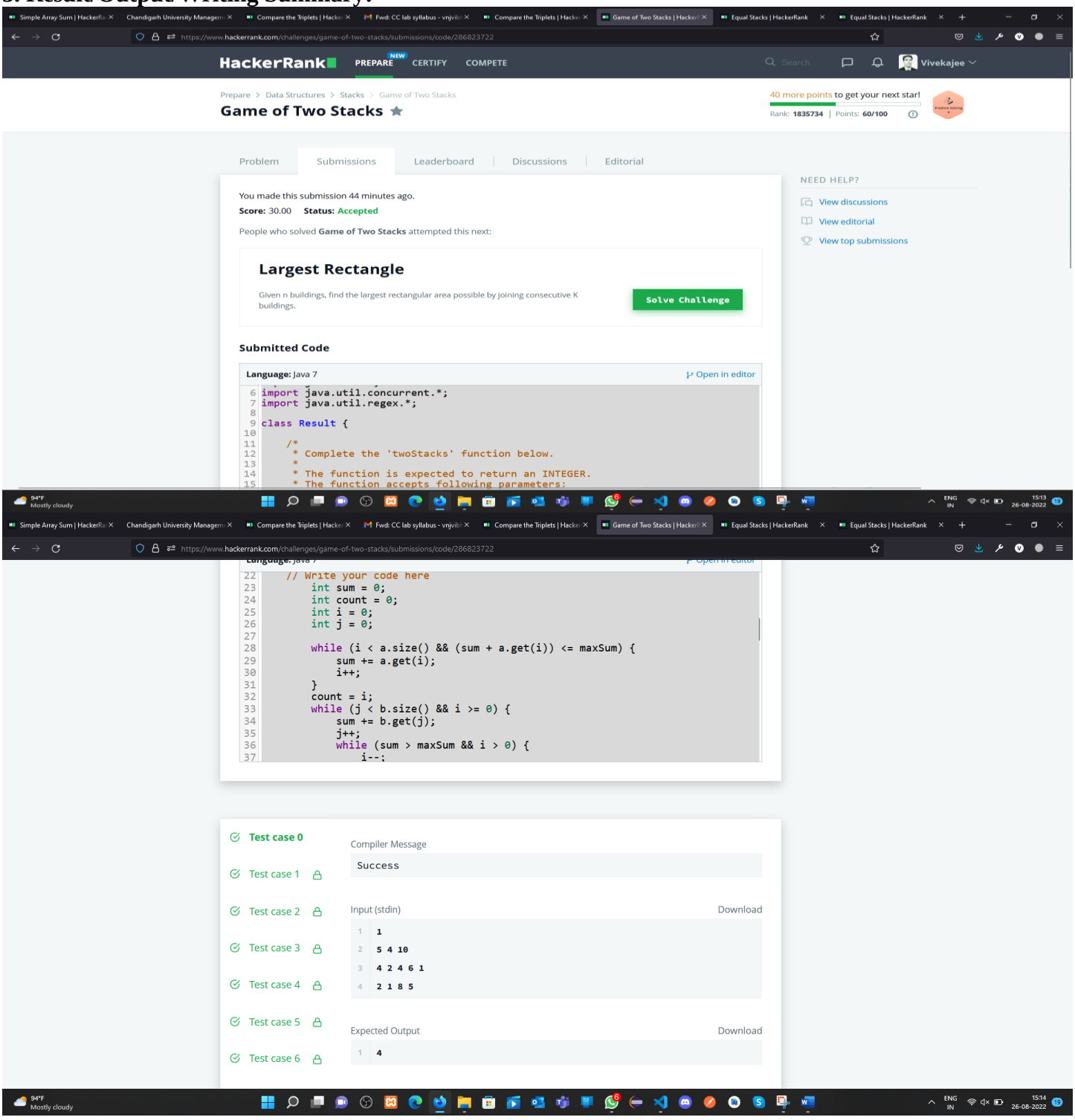
- Laptop or Desktop
- Hacker-Rank Account

4. Steps for experiment/practical/Code:

```
import java.io.*;
import java.math.*;
import java.security.*;
import java.text.*;
import java.util.*;
import java.util.concurrent.*;
import java.util.regex.*;

class Result {

    public static int twoStacks(int maxSum, List<Integer> a, List<Integer> b) {
    // Write your code here
        int sum = 0;
        int count = 0;
        int i = 0;
        int j = 0;

        while (i < a.size() && (sum + a.get(i)) <= maxSum) {
            sum += a.get(i);
            i++;
        }
        count = i;
        while (j < b.size() && i >= 0) {
            sum += b.get(j);
            j++;
            while (sum > maxSum && i > 0) {
                i--;
                sum -= a.get(i);
            }
            if (sum <= maxSum && (i + j) > count)
                count = i + j;
        }

        //System.out.println("count "+count);
        return count;
    }

}
```

```
public class Solution {
    public static void main(String[] args) throws IOException {
        BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(System.in));
        BufferedWriter bufferedWriter = new BufferedWriter(new
FileWriter(System.getenv("OUTPUT_PATH")));

        int g = Integer.parseInt(bufferedReader.readLine().trim());

        for (int gItr = 0; gItr < g; gItr++) {
            String[] firstMultipleInput = bufferedReader.readLine().replaceAll("\\s+$", "").split(" ");

            int n = Integer.parseInt(firstMultipleInput[0]);

            int m = Integer.parseInt(firstMultipleInput[1]);

            int maxSum = Integer.parseInt(firstMultipleInput[2]);

            String[] aTemp = bufferedReader.readLine().replaceAll("\\s+$", "").split(" ");

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

            for (int i = 0; i < n; i++) {
                int aItem = Integer.parseInt(aTemp[i]);
                a.add(aItem);
            }

            String[] bTemp = bufferedReader.readLine().replaceAll("\\s+$", "").split(" ");

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

            for (int i = 0; i < m; i++) {
                int bItem = Integer.parseInt(bTemp[i]);
                b.add(bItem);
            }

            int result = Result.twoStacks(maxSum, a, b);

            bufferedWriter.write(String.valueOf(result));
            bufferedWriter.newLine();
        }

        bufferedReader.close();
        bufferedWriter.close();
    }
}
```

5. Result/Output/Writing Summary:



Down to Zero II:

1. Aim/Overview of the practical:

You are given Q queries. Each query consists of a single number N. You can perform any of the 2 operations on N in each move:

1: If we take 2 integers a and b where $N = a \times b (a \neq 1, b \neq 1)$, then we can change $N = max(a, b)$

2: Decrease the value of N by 1.

Determine the minimum number of moves required to reduce the value of N to 0.

2. Task to be done/ Which logistics used:

Input Format

The first line contains the integer Q.

The next Q lines each contain an integer, N.

Constraints

$1 \leq Q \leq 10^3$

$0 \leq N \leq 10^6$

Output Format

Output Q lines. Each line containing the minimum number of moves required to reduce the value of N to 0.

Sample Input

```
2
3
4
```

Sample Output

```
3
3
```

Explanation

For test case 1, We only have one option that gives the minimum number of moves.

Follow 3 -> 2 -> 1 -> 0. Hence, 3 moves.

For the case 2, we can either go 4 -> 3 -> 2 -> 1 -> 0 or 4 -> 2 -> 1 -> 0. The 2nd option is more optimal. Hence, 3 moves.

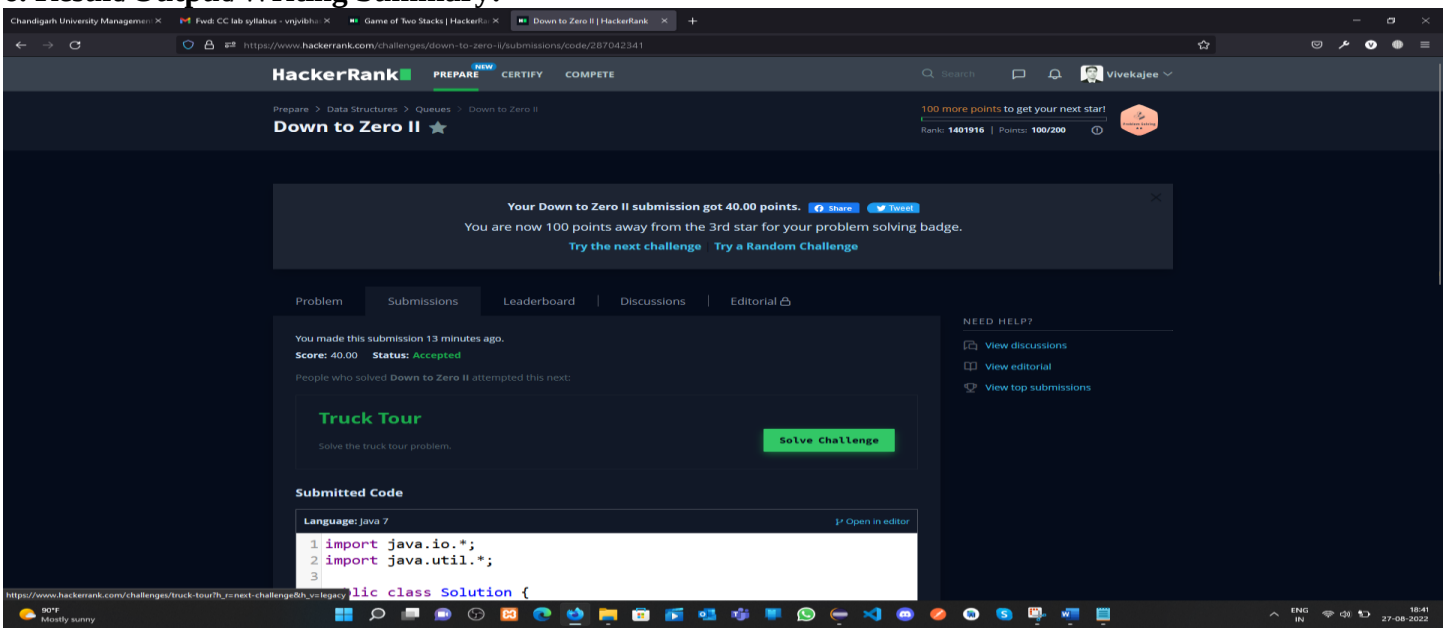
3. Hardware and Software Requirements (For programming-based labs):

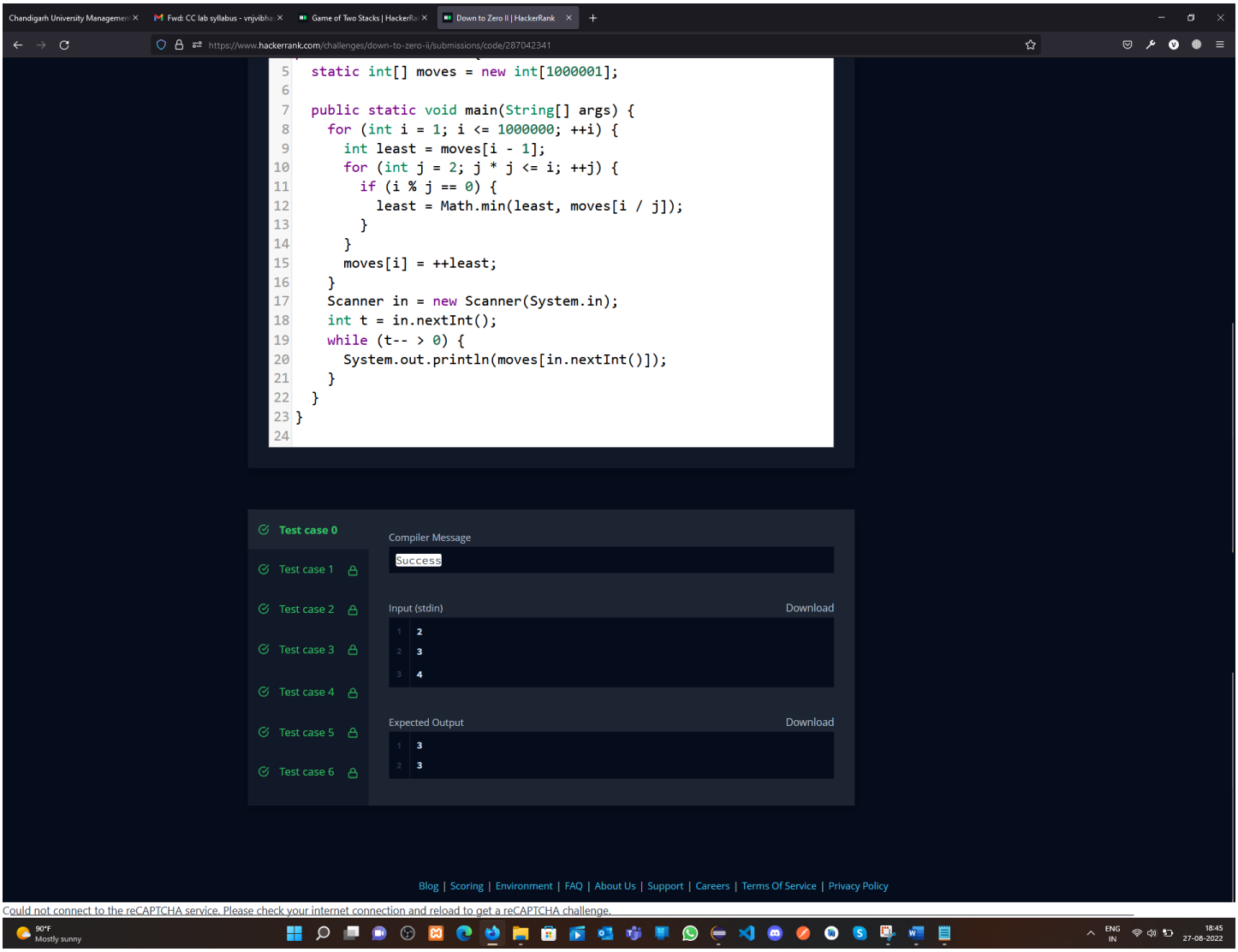
- Laptop or Desktop
- Hacker-Rank Account

4. Steps for experiment/practical/Code:

```
import java.io.*;
import java.util.*;

public class Solution {
  static int[] moves = new int[1000001];

  public static void main(String[] args) {
    for (int i = 1; i <= 1000000; ++i) {
      int least = moves[i - 1];
      for (int j = 2; j * j <= i; ++j) {
        if (i % j == 0) {
          least = Math.min(least, moves[i / j]);
        }
      }
      moves[i] = ++least;
    }
    Scanner in = new Scanner(System.in);
    int t = in.nextInt();
    while (t-- > 0) {
      System.out.println(moves[in.nextInt()]);
    }
  }
}
```

6. Result/Output/Writing Summary:





Learning outcomes (What I have learnt):

1. Concept of Stack and operation**.**
2. Completed my two question.

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.			