

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

**Experiment 2.1**

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

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

**Semester:** 5th **Date of Performance:** 26/08/2022

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

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

A matching pair of brackets is not balanced if the set of brackets it encloses are not matched. For example, {[(])} is not balanced because the contents in between { and } are not balanced. The pair of square brackets encloses a single, unbalanced opening bracket, (, and the pair of parentheses encloses a single, unbalanced closing square bracket, ].

By this logic, we say a sequence of brackets is balanced if the following conditions are met:

• It contains no unmatched brackets.

• The subset of brackets enclosed within the confines of a matched pair of brackets is also a matched pair of brackets

1. **Apparatus / Simulator Used:**

* Windows 7 or above
* Google Chrome

**3. Code:**

#include <bits/stdc++.h>

using namespace std;

int main(){

    int t;

    cin >> t;

    while(t--) {

        string str;

        cin >> str;

        vector<char> v1;

        long long top=-1;

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

            if(str[i]=='{'||str[i]=='['||str[i]=='('||top == -1){

                v1.push\_back(str[i]);

                top++; }

            else if((str[i]=='}' && v1[top]=='{')|| (str[i]==']' && v1  [top]=='[')|| (str[i]==')'&&v1[top]=='(')){

                v1.pop\_back();

                top--;}

            else {

                v1.push\_back(str[i]);

                top++;  }

        if(v1.size() == 0)  {

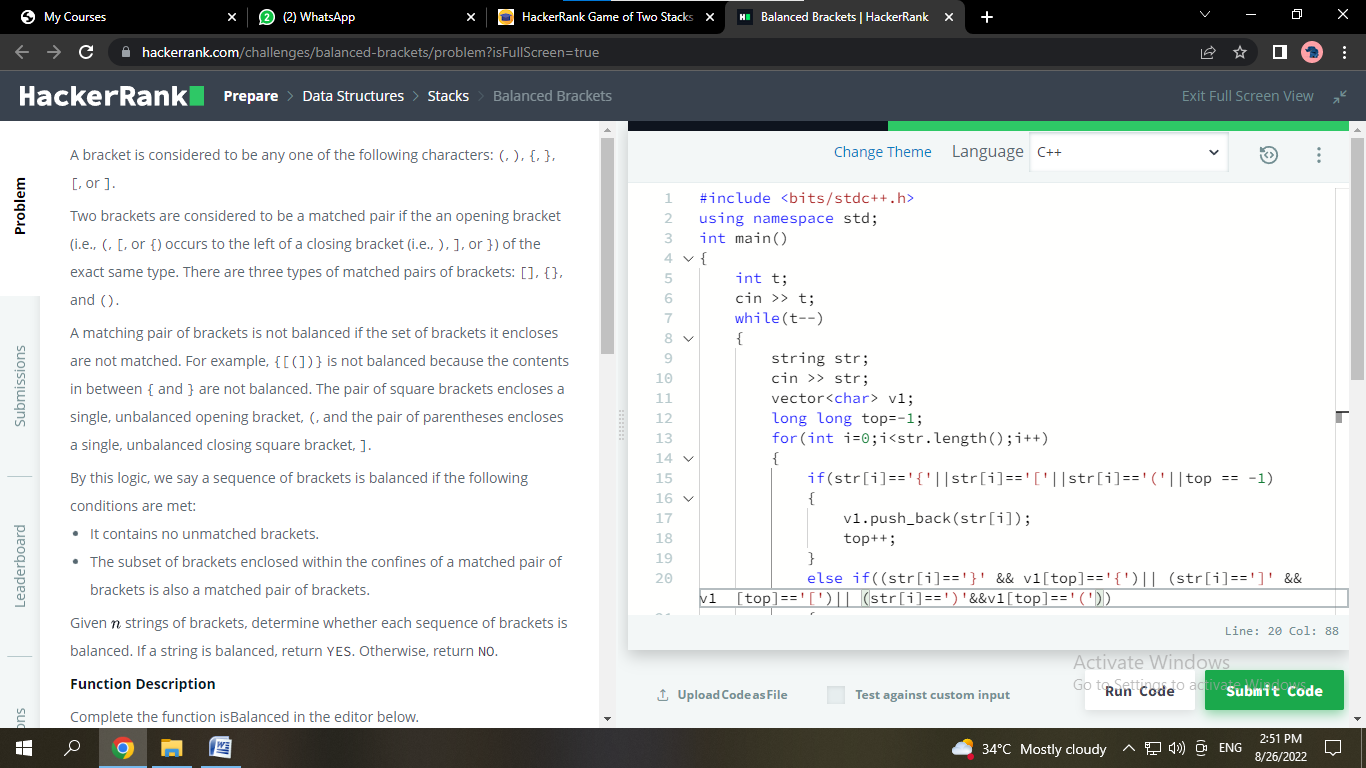
            cout << "YES" << endl; }

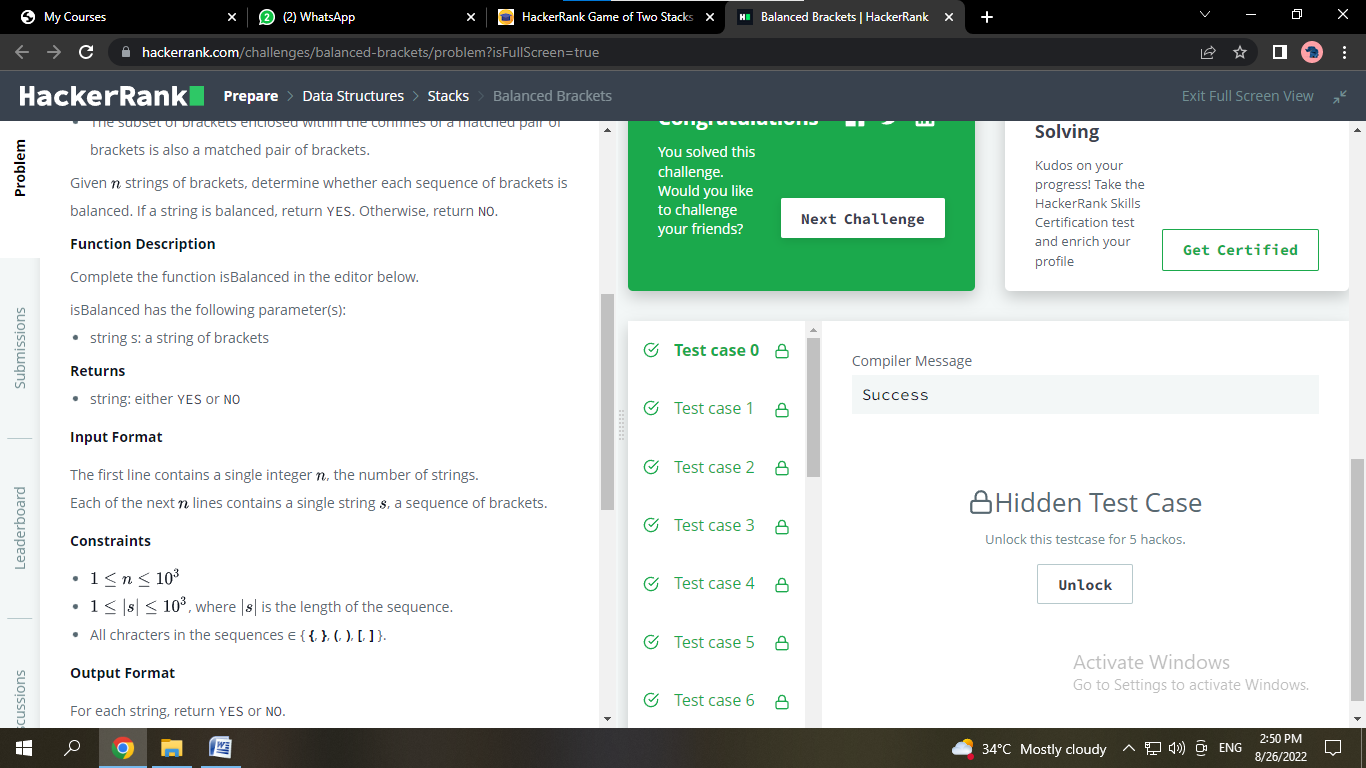
        else cout << "NO" << endl;

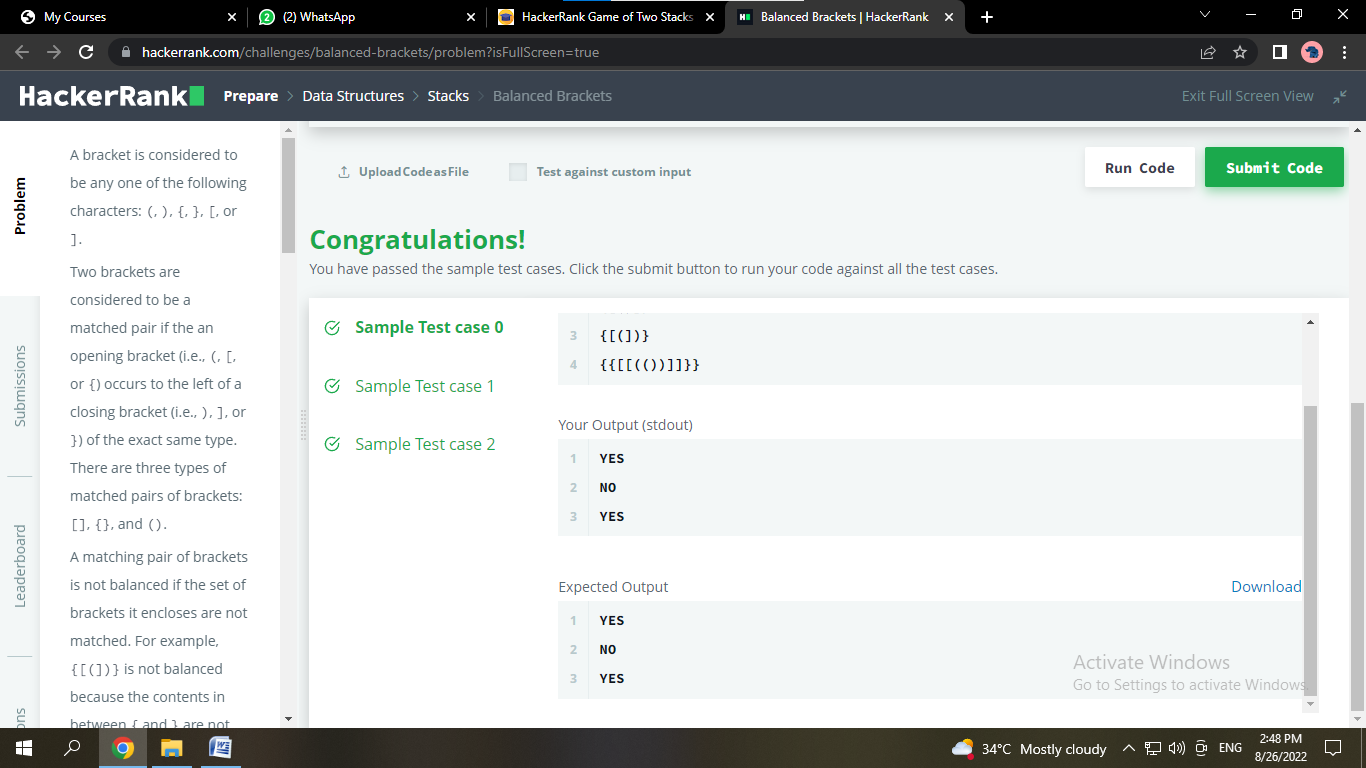
    }

}

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







**Experiment 2.2**

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

Alexa has two stacks of non-negative integers, stack  and stack  where index  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  or stack .
* 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  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 , , and  for  games, find the maximum possible score Nick can achieve.

1. **Apparatus / Simulator Used:**

* Windows 7 or above
* Google Chrome

1. **Code:**

import java.util.\*;

public class GameOfTwoStacks {

    public static void main(String[] args) {

        Scanner in = new Scanner(System.in);

        int g = in.nextInt();

        for(int a0 = 0; a0 < g; a0++){

            int n = in.nextInt();

            int m = in.nextInt();

            int x = in.nextInt();

            long[] a\_sum = new long[n];

            a\_sum[0] = in.nextLong();

            for(int a\_i=1; a\_i < n; a\_i++){

                a\_sum[a\_i] = in.nextLong()+a\_sum[a\_i-1];

            }

            long[] b\_sum = new long[m];

            b\_sum[0] = in.nextLong();

            for(int b\_i=1; b\_i < m; b\_i++){

                b\_sum[b\_i] = in.nextLong()+b\_sum[b\_i-1];

            }

            int ai = a\_sum.length-1;

            int bi = 0;

            int max\_score = 0;

            int score = 0;

            while(ai>=0&&a\_sum[ai] > x)

                ai--;

            if(ai>=0)

                while(bi<b\_sum.length&&a\_sum[ai]+b\_sum[bi]<=x)

                    bi++;

            else

                while(bi<b\_sum.length&&b\_sum[bi]<=x)

                    bi++;

            for(ai = ai; ai >= -1; ai--)

            {

                if(ai>=0)

                {

                    while(bi<b\_sum.length&&a\_sum[ai]+b\_sum[bi]<=x)

                        bi++;

                    score = ai+bi+1;

                }

                else

                {

                    while(bi<b\_sum.length&&b\_sum[bi]<=x)

                        bi++;

                    score = bi;

                }

                if(score>max\_score)

                    max\_score = score;

            }

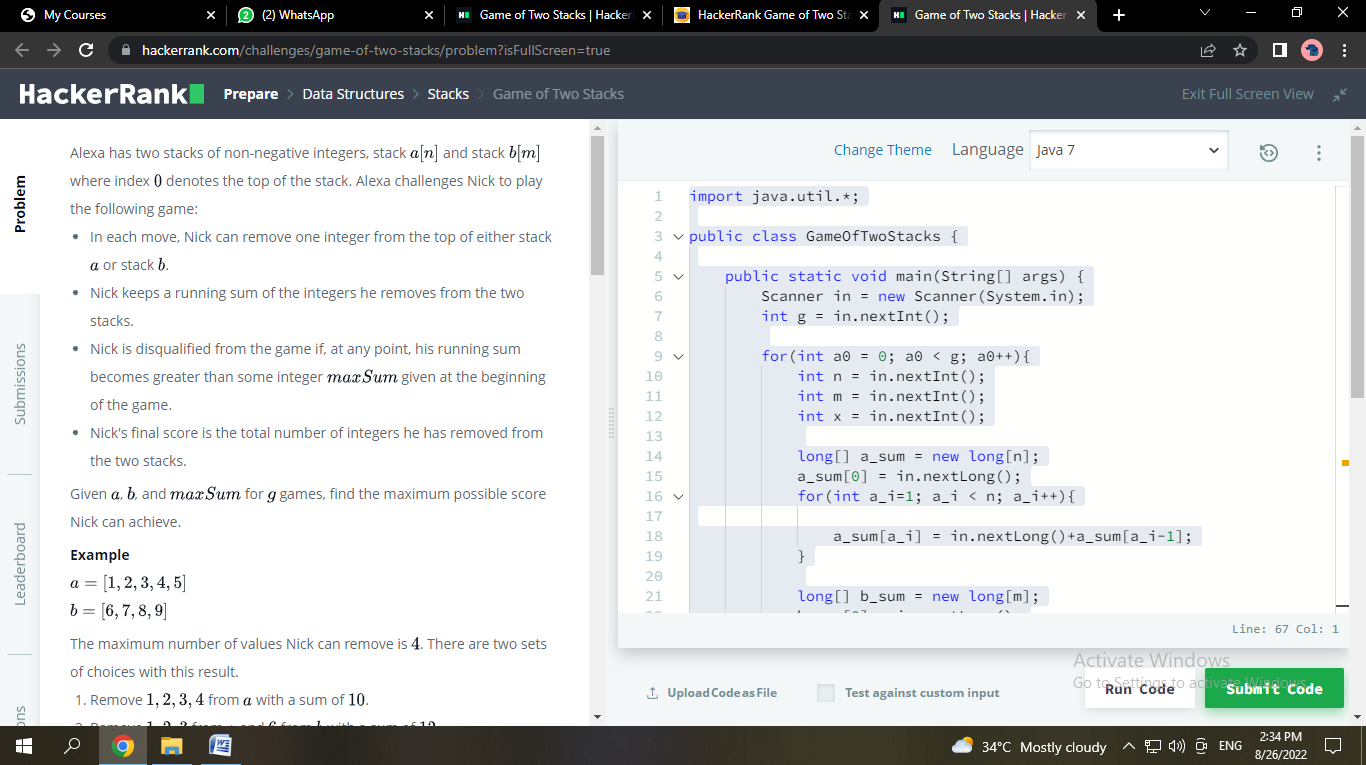
            System.out.println(max\_score);

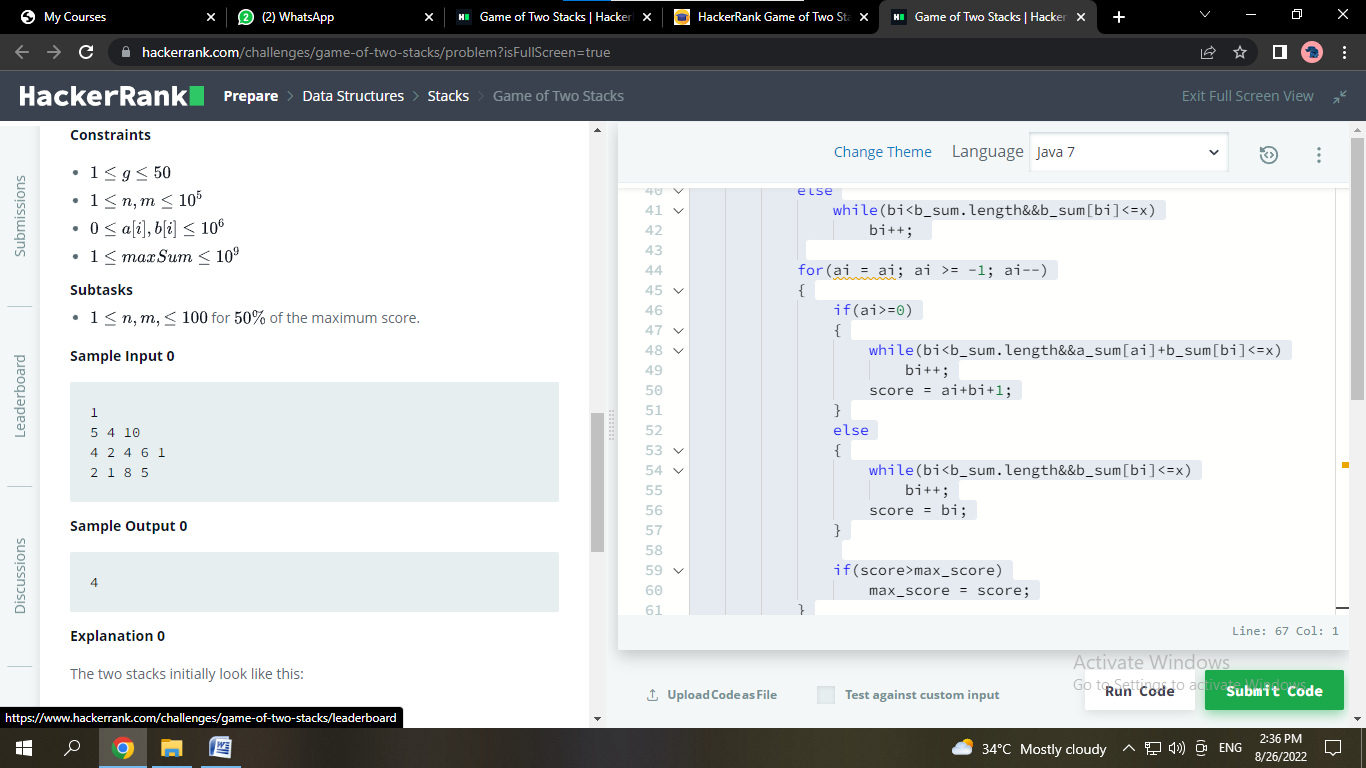
        }

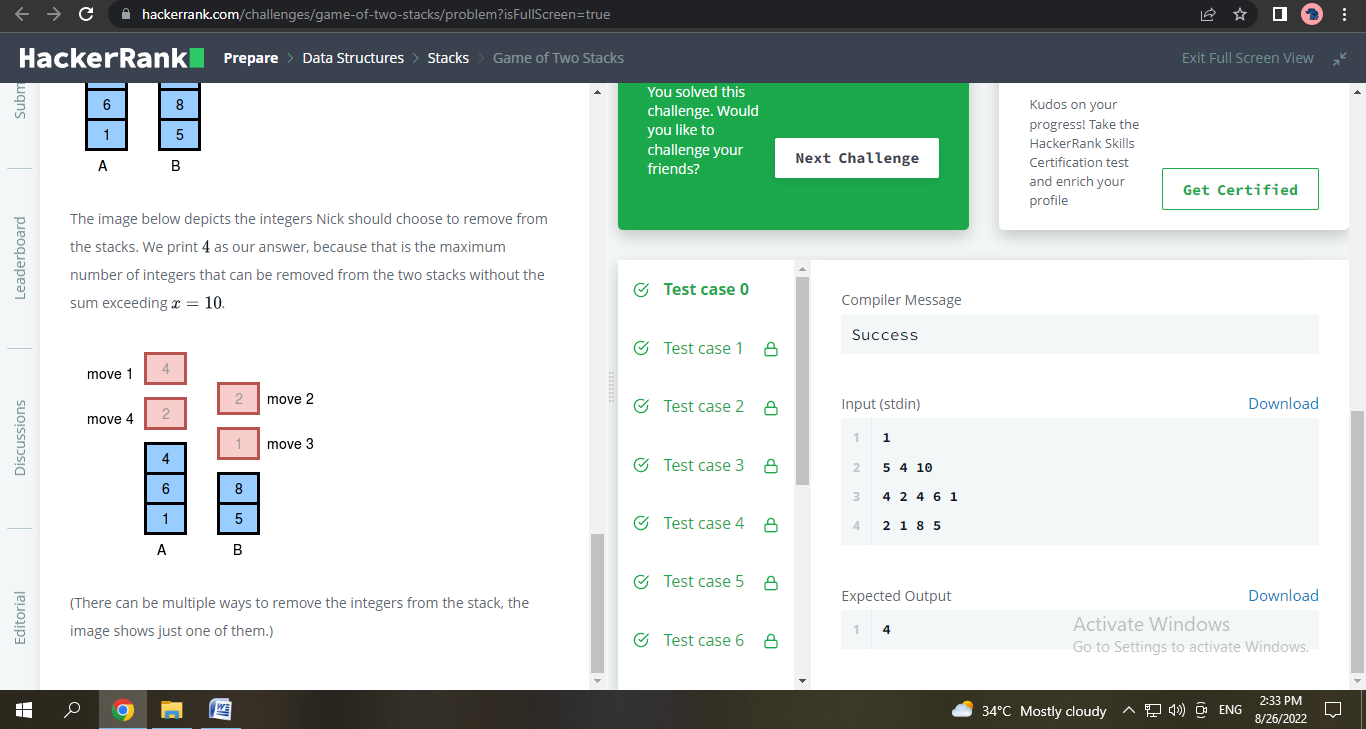
    }

}

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







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

1. Learnt about Array Stack data Structure
2. Learnt different approaches to equalize the stack.

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