**“Experiment 1.2”**

Student Name: **SUMIT KUMAR** UID: **20BCS8226**

Branch: **CSE** Section/Group: **MM-808-A**

Semester: **05** Date of Submission: **05/09/22**

Subject Name: **Competitive Coding Lab** Subject Code: **20CSP-314**

**Aim/Overview of the practical:**

**Question 1: Equal Stacks**

You have three stacks of cylinders where each cylinder has the same diameter, but they may vary in height. You can change the height of a stack by removing and discarding its topmost cylinder any number of times.

Find the maximum possible height of the stacks such that all of the stacks are exactly the same height. This means you must remove zero or more cylinders from the top of zero or more of the three stacks until they are all the same height, then return the height.

**Question 2: Game of Two Stacks**

Alexa has two stacks of non-negative integers, stack a[n] and stack b[m] 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 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 maxSum 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, maxSum and for g games, find the maximum possible score Nick can achieve.

**Algorithm/Flowchart (For programming based labs):**

**Question 1:**

Declare Height and Lengths

for(auto val : h1){

        \_heightH1 += val;

    }

    for(auto val : h2){

        \_heightH2 += val;

    }

    for(auto val : h3){

        \_heightH3 += val;

    }

Begin while

if((\_heightH1 == \_heightH2) && (\_heightH1 == \_heightH3)) break;

        if (\_heightH1 >= \_heightH2 && \_heightH1 >= \_heightH3){

            \_heightH1 -= h1[\_LengthH1];

            ++\_LengthH1;

        }

        else if (\_heightH2 >= \_heightH1 && \_heightH2 >= \_heightH3){

            \_heightH2 -= h2[\_LengthH2];

            ++\_LengthH2;

        }

        else {

            \_heightH3 -= h3[\_LengthH3];

            ++\_LengthH3;

        }

End while

**Question 2:**

Declare main method

Begin While

while (n--) {

    int a\_i, b\_i, max\_sum;

       cin >> a\_i >> b\_i >> max\_sum;

       int val[a\_i+b\_i];

begin loops

for (int i = a\_i-1; i >= 0; i--) {

       cin >> val[i];

    }

    for (int i = a\_i; i < a\_i+b\_i; i++) {

         cin >> val[i];

     }

End loop

End while

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

**Question 1 Code:**

#include <bits/stdc++.h>

using namespace std;

string ltrim(const string &);

string rtrim(const string &);

vector<string> split(const string &);

int equalStacks(vector<int> h1, vector<int> h2, vector<int> h3) {

    int \_LengthH1 = 0;

    int \_LengthH2 = 0;

    int \_LengthH3 = 0;

    int \_heightH1 = 0;

    int \_heightH2 = 0;

    int \_heightH3 = 0;

    for(auto val : h1){

        \_heightH1 += val;

    }

    for(auto val : h2){

        \_heightH2 += val;

    }

    for(auto val : h3){

        \_heightH3 += val;

    }

    while (1){

        if((\_heightH1 == \_heightH2) && (\_heightH1 == \_heightH3)) break;

        if (\_heightH1 >= \_heightH2 && \_heightH1 >= \_heightH3){

            \_heightH1 -= h1[\_LengthH1];

            ++\_LengthH1;

        }

        else if (\_heightH2 >= \_heightH1 && \_heightH2 >= \_heightH3){

            \_heightH2 -= h2[\_LengthH2];

            ++\_LengthH2;

        }

        else {

            \_heightH3 -= h3[\_LengthH3];

            ++\_LengthH3;

        }

    }

    return \_heightH1;

}

int main()

{

    ofstream fout(getenv("OUTPUT\_PATH"));

    string first\_multiple\_input\_temp;

    getline(cin, first\_multiple\_input\_temp);

    vector<string> first\_multiple\_input = split(rtrim(first\_multiple\_input\_temp));

    int n1 = stoi(first\_multiple\_input[0]);

    int n2 = stoi(first\_multiple\_input[1]);

    int n3 = stoi(first\_multiple\_input[2]);

    string h1\_temp\_temp;

    getline(cin, h1\_temp\_temp);

    vector<string> h1\_temp = split(rtrim(h1\_temp\_temp));

    vector<int> h1(n1);

    for (int i = 0; i < n1; i++) {

        int h1\_item = stoi(h1\_temp[i]);

        h1[i] = h1\_item;

    }

    string h2\_temp\_temp;

    getline(cin, h2\_temp\_temp);

    vector<string> h2\_temp = split(rtrim(h2\_temp\_temp));

    vector<int> h2(n2);

    for (int i = 0; i < n2; i++) {

        int h2\_item = stoi(h2\_temp[i]);

        h2[i] = h2\_item;

    }

    string h3\_temp\_temp;

    getline(cin, h3\_temp\_temp);

    vector<string> h3\_temp = split(rtrim(h3\_temp\_temp));

    vector<int> h3(n3);

    for (int i = 0; i < n3; i++) {

        int h3\_item = stoi(h3\_temp[i]);

        h3[i] = h3\_item;

    }

    int result = equalStacks(h1, h2, h3);

    fout << result << "\n";

    fout.close();

    return 0;

}

string ltrim(const string &str) {

    string s(str);

    s.erase(

        s.begin(),

        find\_if(s.begin(), s.end(), not1(ptr\_fun<int, int>(isspace)))

    );

    return s;

}

string rtrim(const string &str) {

    string s(str);

    s.erase(

        find\_if(s.rbegin(), s.rend(), not1(ptr\_fun<int, int>(isspace))).base(),

        s.end()

    );

    return s;

}

vector<string> split(const string &str) {

    vector<string> tokens;

    string::size\_type start = 0;

    string::size\_type end = 0;

    while ((end = str.find(" ", start)) != string::npos) {

        tokens.push\_back(str.substr(start, end - start));

        start = end + 1;

    }

    tokens.push\_back(str.substr(start));

    return tokens;

}

**Question 2 Code:**

#include <bits/stdc++.h>

using namespace std;

#include <bits/stdc++.h>

using namespace std;

int main() {

    int n;

    cin >> n;

    while (n--) {

        int a\_i, b\_i, max\_sum;

        cin >> a\_i >> b\_i >> max\_sum;

       int val[a\_i+b\_i];

       for (int i = a\_i-1; i >= 0; i--) {

           cin >> val[i];

       }

       for (int i = a\_i; i < a\_i+b\_i; i++) {

           cin >> val[i];

       }

       int least\_a = 0, init\_sum = 0;

       while (max\_sum >= init\_sum + val[(a\_i-1)-least\_a] && least\_a < a\_i)

{

           init\_sum += val[(a\_i-1)-least\_a];

           least\_a ++;

       }

       int b\_up = a\_i;

       int a\_up = a\_i-least\_a;

       int max = least\_a;

       while (a\_up <= a\_i && b\_up < a\_i + b\_i) {

        int b\_sum = 0;

           while (max\_sum - init\_sum >= b\_sum + val[b\_up] && b\_up < a\_i + b\_i) {

           b\_sum += val[b\_up];

           b\_up++;

       }

       if (b\_up - a\_up > max) max = b\_up - a\_up;

       init\_sum -= val[a\_up++];

       init\_sum += b\_sum;

       }

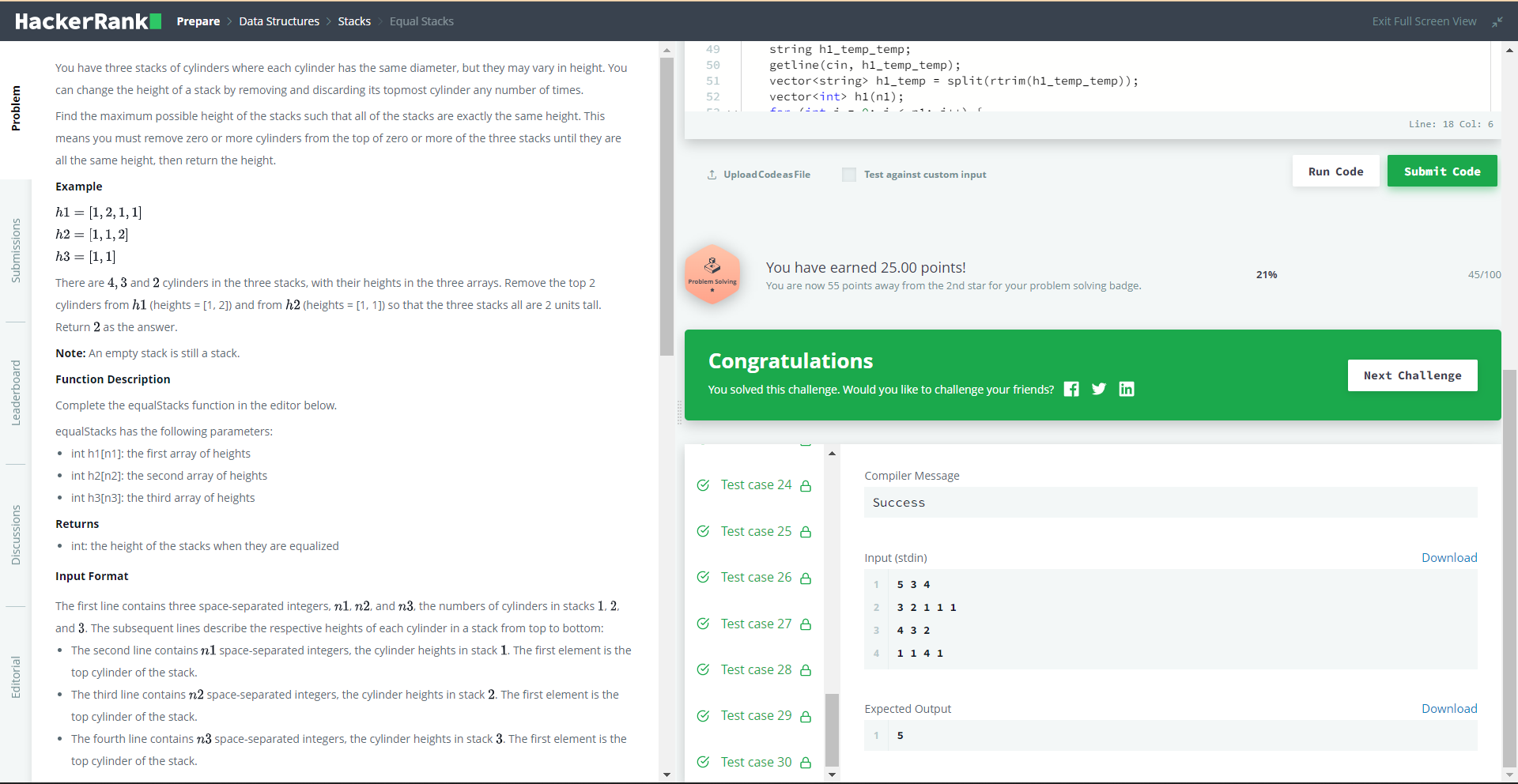
       cout << max << "\n";

    }

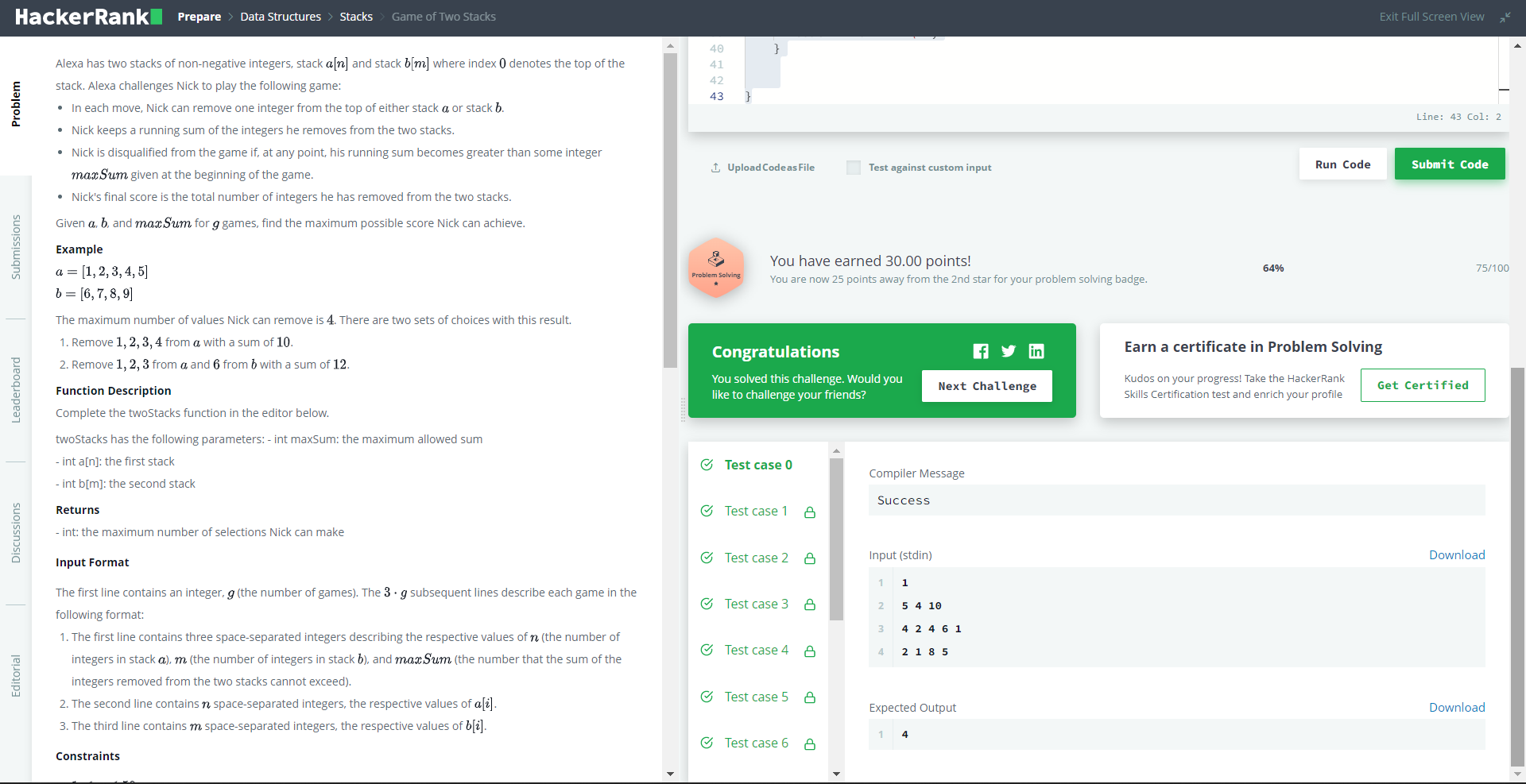
}

**Result/Output/Writing Summary:**

**Question 1 Output:**



**Question 2 Output:**



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

1. **Implementation of stacks.**

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