**“Experiment 1.1”**

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**

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

**Question 1:**

Given an array of integers, find the sum of its elements.

For example, if the array , , so return .

Function Description

Complete the simpleArraySum function in the editor below. It must return the sum of the array elements as an integer.

simpleArraySum has the following parameter(s):

ar: an array of integers.

**Question 2:**

Given a square matrix, calculate the absolute difference between the sums of its diagonals.

For example, the square matrix  is shown below:

1 2 3

4 5 6

9 8 9

The left-to-right diagonal = 1+5+9=15. The right to left diagonal = 3+5+9=17. Their absolute difference is|15-17|=2 .

Function description

Complete the function in the editor below.

diagonalDifference takes the following parameter:

int arr[n][m]: an array of integers

Return

int: the absolute diagonal difference.

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

**Question 1:**

Define Sum=0;

Begin Loop upto ar.size;

Calculate Sum;

End loop;

Return sum;

**Question 2:**

Define ld=0,rd=0;

Begin Loop

Begin Another Loop

If(i==j)

ld+=ar[i][j]

If(i+j==ar.size()-1)

rd+=ar[i][j]

end if

end loop

end loop

return abs(ld-rd);

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

**Question 1:**

#include <bits/stdc++.h>

using namespace std;

string ltrim(const string &);

string rtrim(const string &);

vector<string> split(const string &);

int simpleArraySum(vector<int> ar) {

    int sum=0;

    for(int i=0;i<ar.size();i++){

        sum=sum+ar.at(i);

    }

    return sum;

}

int main()

{

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

    string ar\_count\_temp;

    getline(cin, ar\_count\_temp);

    int ar\_count = stoi(ltrim(rtrim(ar\_count\_temp)));

    string ar\_temp\_temp;

    getline(cin, ar\_temp\_temp);

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

    vector<int> ar(ar\_count);

    for (int i = 0; i < ar\_count; i++) {

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

        ar[i] = ar\_item;

    }

    int result = simpleArraySum(ar);

    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:**

#include <bits/stdc++.h>

using namespace std;

string ltrim(const string &);

string rtrim(const string &);

vector<string> split(const string &);

int diagonalDifference(vector<vector<int>> arr) {

    int rd,ld;

    ld=0;

    rd=0;

    for(int i=0;i<arr.size();i++){

        for(int j=0;j<arr.size();j++){

            if(i==j){

                ld+=arr[i][j];

            }

            if(i+j==arr.size()-1){

                rd+=arr[i][j];

            }

        }

    }

    return abs(ld-rd);

}

int main()

{

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

    string n\_temp;

    getline(cin, n\_temp);

    int n = stoi(ltrim(rtrim(n\_temp)));

    vector<vector<int>> arr(n);

    for (int i = 0; i < n; i++) {

        arr[i].resize(n);

        string arr\_row\_temp\_temp;

        getline(cin, arr\_row\_temp\_temp);

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

        for (int j = 0; j < n; j++) {

            int arr\_row\_item = stoi(arr\_row\_temp[j]);

            arr[i][j] = arr\_row\_item;

        }

    }

    int result = diagonalDifference(arr);

    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;

    }

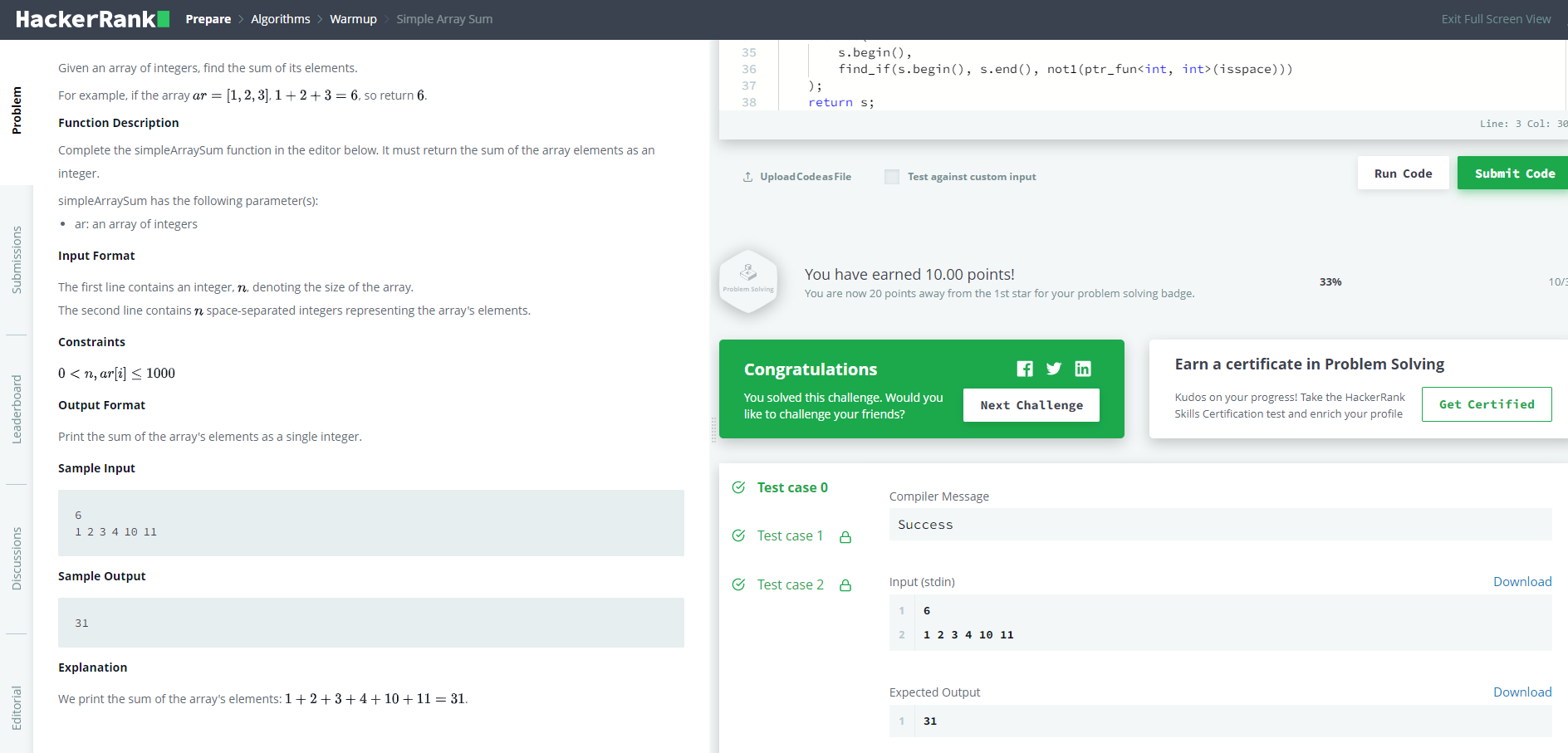
    tokens.push\_back(str.substr(start));

    return tokens;

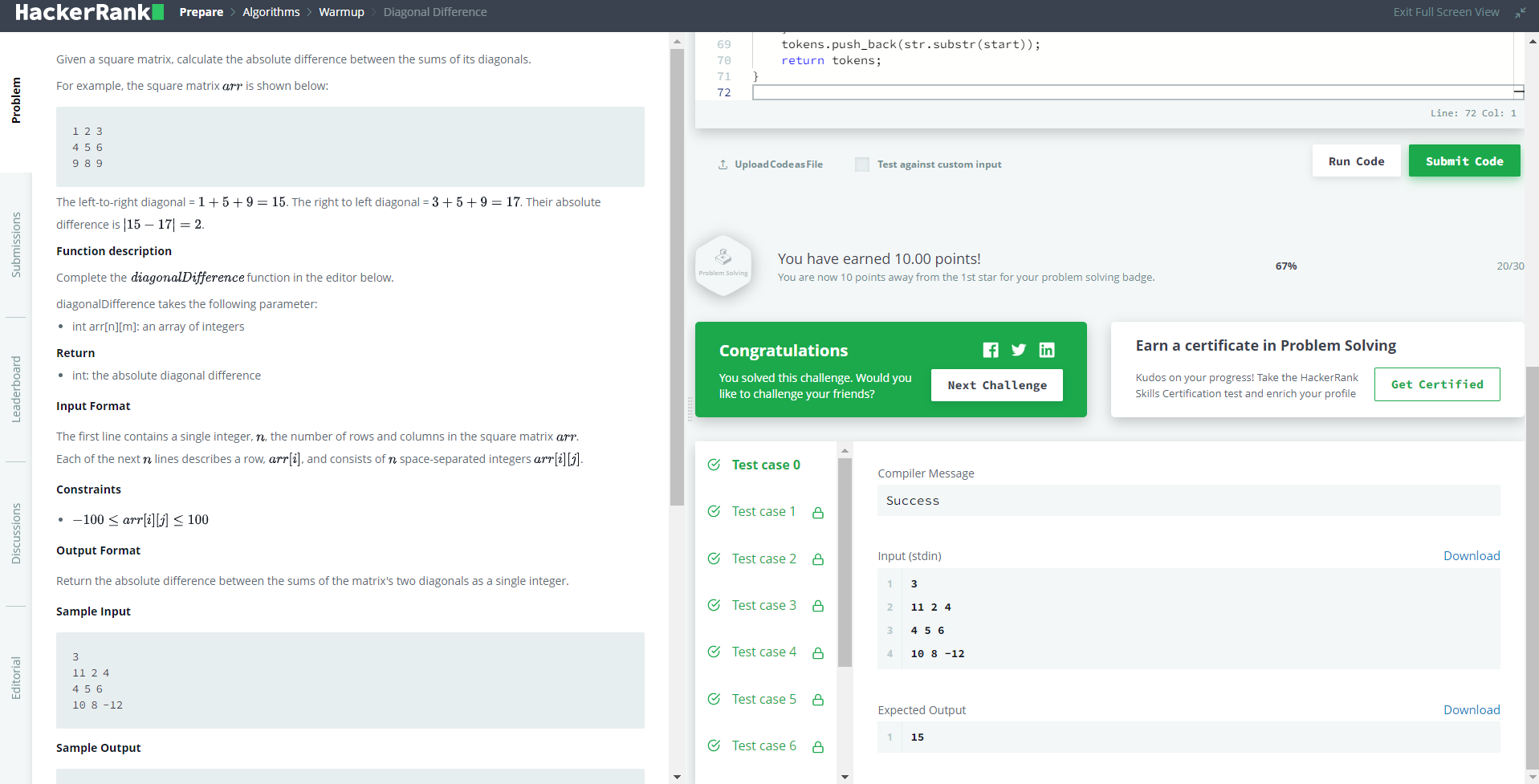
}

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

**Question 1 Output:**



**Question 2 Output:**



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

1. The concept of Array Indexing is cleared.

2. Concept of Array Addition.

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