Experiment 2

Student Name: Anjali Singh **UID: 20BCS9239**
Branch: CSE **Section/Group:607/A**
Semester: 5th **Date of Performance:04/09/22**
Subject Name: CC Lab **Subject Code: 21CSP-314**

1. **Aim/Overview of the practical:** To implement the concept of STACK & QUEUES.

2. **Task to be done/ Which logistics used:**

In this practical we are going understand various problems and find out better approach to solve particular problem related to stack and queues on hackerrank.

a

A bracket is considered to be any one of the following characters: (,), {, }, [, or].

Two brackets are considered to be a matched pair if the an opening bracket (i.e., (, [, or {) occurs to the left of a closing bracket (i.e.,),], or }) of the exact same type. There are three types of matched pairs of brackets:
[], {}, and ().

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.
Given strings of brackets, determine whether each sequence of brackets is balanced. If a string is balanced, return YES. Otherwise, return NO.

b) Game of two stacks

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.

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.

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

3. Algorithm/Flowchart (For programming based labs):

a) Balanced Brackets

- START
- Declare a vector of char type named as brackets and a bool variable ans =true.
- Run a for loop and check for every open bracket is there a close bracket or not.
- If True return yes Else return No.
- END

b) Game of two stacks

- START
- Declare sum and count two variables of int type.
- Run a while loop to traverse vector a and if store elements in sum till sum<maxSum.
- Run another while loop to do same thing with vector b and count no of elements stored in sum by using count pointer.
- Return count.
- END

4. Steps for experiment/practical/Code:

A) Balanced Brackets

```
string isBalanced(string s)
{
    vector<char> brackets;

    bool ans = true;

    for (char c : s)
    {
        if (c == ')' || c == ']' || c == '}')
        {
```

```
            if ((!brackets.empty()) && ((c == ')' && brackets.back() == '(')
|| (c == ']' && brackets.back() == '[') || (c == '}' && brackets.back() ==
'{')))
            {
                brackets.pop_back();
            }
            else
            {
                ans = false;
                break;
            }
        }
        else
            brackets.push_back(c);
    }
    if (ans && brackets.empty())
        return "YES";
    else
        return "NO";
}
```

B) Game of two stacks .

```
#include <bits/stdc++.h>
using namespace std;

int main()
{
    int n, i, j;
    cin >> n;
    int arr[n][n];
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n; j++)
```

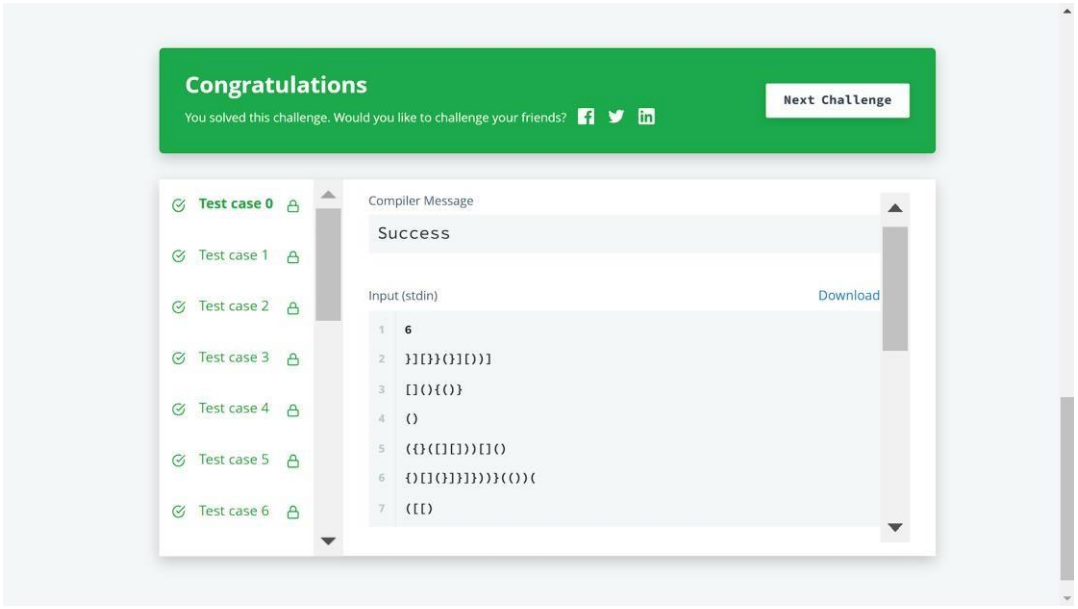
```
        {
            cin >> arr[i][j];
        }
    }
    int sumLeftToRight = 0, sumRightToLeft = 0;
    for (int i = 0; i < n; i++)
    {
        sumLeftToRight += arr[i][i];
        sumRightToLeft += arr[i][n - i - 1];
    }
    int res = abs(sumRightToLeft - sumLeftToRight);
    cout << res << endl;
    return 0;
}
```

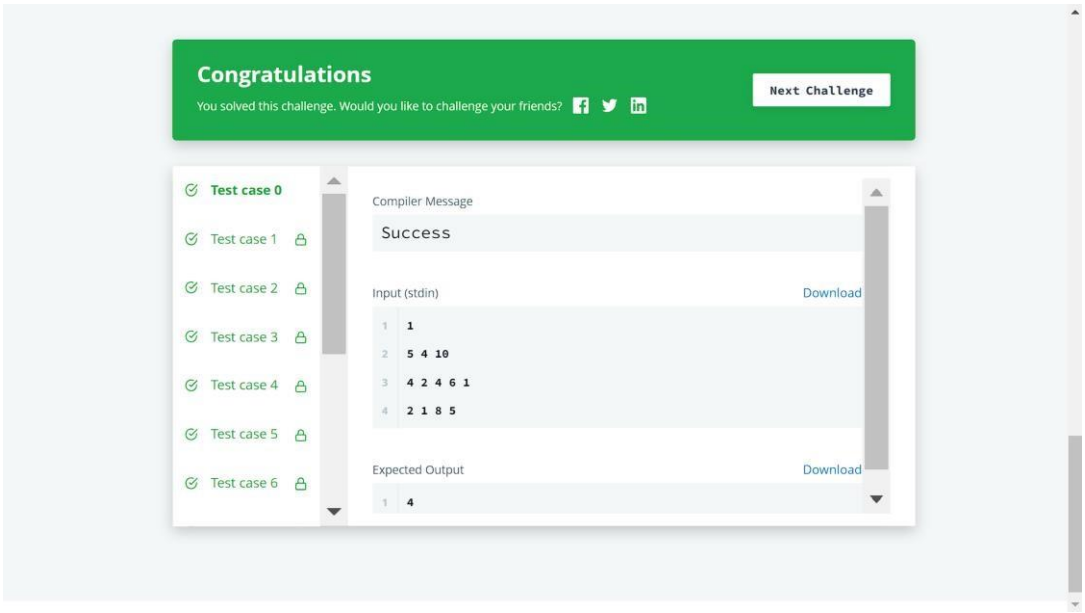
5. Observations/Discussions/ Complexity Analysis:

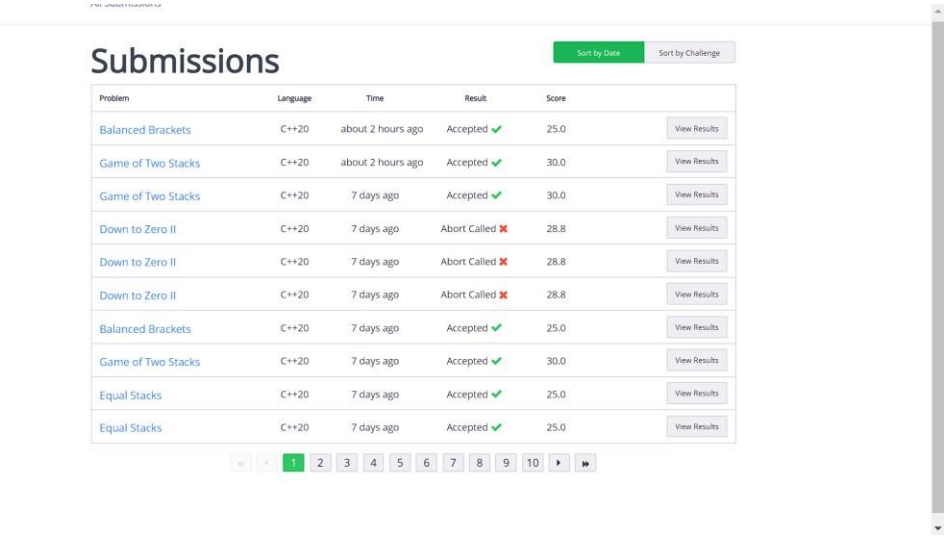
a). In balance brackets function all we trying to do is to check that all opening brackets should have a close bracket and if this condition is not met then the string is not balance otherwise it is balance here we are using vector for its implementation.

b) In game of two stacks function we are trying to play a kind of game with two stacks with some elements in such a way that, we have to remove elements from both stacks until the some of removed element is less than maxSum . In this we have to remove elements from both stacks. Then just count the total numbers of elements removed and store it in a variable name count and in the last of the program just return the count variable.

6. Result/Output/Writing Summary:


Congratulations
You solved this challenge. Would you like to challenge your friends?
Next Challenge
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6
Compiler Message
Success
Input (stdin)
Download
6
}][]}(}][)]
[](){()}
()
({}([][]))[]()
{}[](}]}}}}}(())(
([])


Congratulations
You solved this challenge. Would you like to challenge your friends?
Next Challenge
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6
Compiler Message
Success
Input (stdin)
Download
1
5 4 10
4 2 4 6 1
2 1 8 5
Expected Output
Download
4



Learning outcomes (What I have learnt):

1. I have learnt how to use different functions and library of c++.
2. I have learnt how to deal with real time problems.
3. Both questions helps me to build different logic and concept.
4. Learnt how to implement stacks and do various types of functions with it.

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.			