Experiment 2

Student Name: UID:

Branch:BE-CSE Section/Group:

Semester:5th DateofPerformance:

Subject Name: AP Subject Code: 22CSP-314

Problem 1

1. Aim: Equal Stacks

2. Objectives: To make all three stacks the same height, with the maximum possible height that can be achieved.

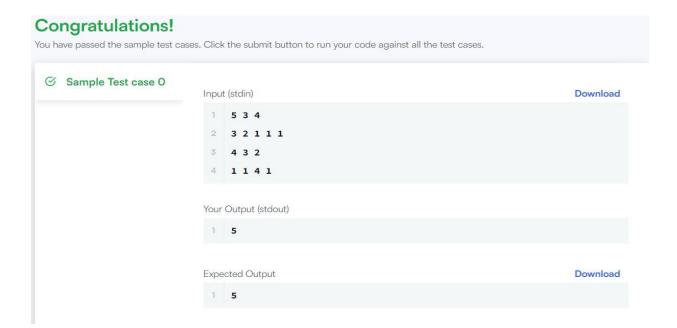
3. Code:

```
#include<iostream>
using namespace std;
int main(){
  int n1;
  int n2;
  int n3;
  cin >> n1 >> n2 >> n3;
  int h1 = 0, h2 = 0, h3 = 0; // heights of the 3 stacks
  vector<int> tower1(n1);
  for(int i = 0; i < n1; i++){
    cin>>tower1[i];
    h1 += tower1[i];
  }
  vector<int> tower2(n2);
  for(int i = 0; i < n2; i++){
    cin>>tower2[i];
```

```
h2 += tower2[i];
           vector<int> tower3(n3);
          for(int i = 0; i < n3; i++){
                    cin>>tower3[i];
                    h3 += tower3[i];
          // Use a greedy approach, always remove cylinders from the tallest tower until al
1 towers
          // have the same height.
           bool equalHeight = false;
          if(h1 == h2 \&\& h2 == h3) {
                      equalHeight = true;
            }
           int r1 = 0, r2 = 0, r3 = 0; // Store the indices of which cylinder to remove
           while(!equalHeight) {
                      if(h1 >= h2 \&\& h1 >= h3) {
                                  h1 = tower1[r1];
                                  r1++;
                        ext{less if (h2 >= h1 \&\& h2 >= h3) {}}
                                  h2 = tower2[r2];
                                  r2++;
                        ellet elle
                                  h3 = tower3[r3];
```

```
r3++;
}
if((h1 == h2 && h2 == h3) || (h1 == 0 && h2 == 0 && h3 == 0)) {
    equalHeight = true;
}
cout<<h1;
return 0;
}
```

4. Output:



Problem 2

- 1.Aim: Balanced Brackets
- **2.Objective:** Write a function isBalanced(s) that takes a string s of brackets and returns "YES" if the sequence is balanced, otherwise returns "NO"

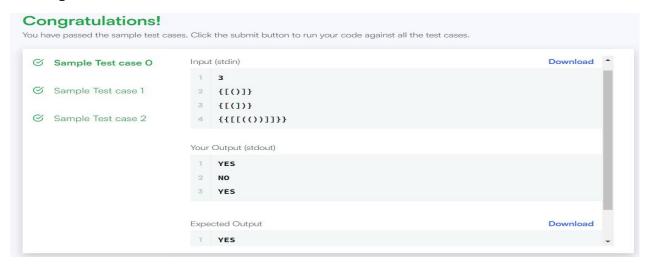
3. Code:

```
#include <bits/stdc++.h>
using namespace std;
string ltrim(const string &);
string rtrim(const string &);
/*
* Complete the 'isBalanced' function below.
* The function is expected to return a STRING.
* The function accepts STRING s as parameter.
*/
string isBalanced(string s) {
stack<char> stk;
  unordered_map<char, char> matching_bracket = { { ')', '('}, { ']', '['}, { '}', '{ '}};
  for (char ch: s) {
     if (ch == '(' || ch == '[' || ch == '{'}) {
       stk.push(ch);
     } else {
```

```
if (stk.empty() || stk.top() != matching_bracket[ch]) {
          return "NO";
       stk.pop();
     }
    return stk.empty() ? "YES" : "NO";
}
int main()
{
  ofstream fout(getenv("OUTPUT_PATH"));
string t_temp;
  getline(cin, t_temp);
int t = stoi(ltrim(rtrim(t_temp)));
for (int t_itr = 0; t_itr < t; t_itr++) {
     string s;
     getline(cin, s);
string result = isBalanced(s);
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;}
```

4. Output:



5. Learning outcomes:

- a) Understanding Stack Basics: Learn what a stack is and how it operates (LIFO Last In, First Out).
- b) Finding Equal Heights: Understand the process of comparing and adjusting the heights of different stacks.
- c) Real-World Applications: Understand the importance of balanced brackets in programming and other fields.