CHANDIGARH UNIVERSITY
UNIVERSITY INSTITUTE OF ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Submitted By: Satyam **Submitted To: Navneet Chaudhry**	
Subject Name	Competitive Coding
Subject Code	21 CSP-314
Branch	BE-CSE
Semester	5th

LAB INDEX

NAME: Satyam
SUBJECT NAME: Competitive Coding Lab
UID: 20BCS9393
SUBJECT CODE: 21CSP-314
SECTION: 607-A

Sr. No	Program	Date	Evaluation				Sign
			LW (12)	VV (8)	FW (10)	Total (30)	
1.	**ARRAYS:** https://www.hackerrank.com/challenges/30-arrays/problem https://www.hackerrank.com/challenges/simple-array-sum/problem?isFullScreen=true https://www.hackerrank.com/challenges/compare-the-triplets/problem?isFullScreen=true https://www.hackerrank.com/challenges/diagonal-difference/problem?isFullScreen=true	04-Aug-2022					
2.	**STACK & QUEUES:** https://www.hackerrank.com/challenges/equal-stacks/problem?isFullScreen=true https://www.hackerrank.com/challenges/game-of-two-stacks/problem?isFullScreen=true https://www.hackerrank.com/challenges/balanced-brackets/problem?isFullScreen=true https://www.hackerrank.com/challenges/down-to-zero-ii/problem?isFullScreen=true https://www.hackerrank.com/challenges/truck-tour/problem?isFullScreen=true	18-Aug-2022					

EXPERIMENT-2(a)

1. Aim/Overview of the practical:

To demonstrate the concept of Stack and Queue

2. Task to be done/ Which logistics used: https://www.hackerrank.com/challenges/equal-stacks/problem?isFullScreen=true

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.

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

1. We'll first find the sum of all elements of array h1,h2,h3 respectively and will store them in sum1, sum2,sum3 variable respectively.

2. Now iterate through all three array by single iteration till sum1, sum2 and sum3 don't get equal.

3. While they are not equal, find the max value from them and store it in temp variable.

4.if temp == sum1 then subtract element from ith position in h1 array from sum1. And increase I.

5. Else if temp == sum2 then subtract element from jth position in h2 array from sum2. And increase j.

6. Else subtract element from kth position in h1 array from h3. And increase k.

7.return sum1.

Steps for experiment/practical/Code:

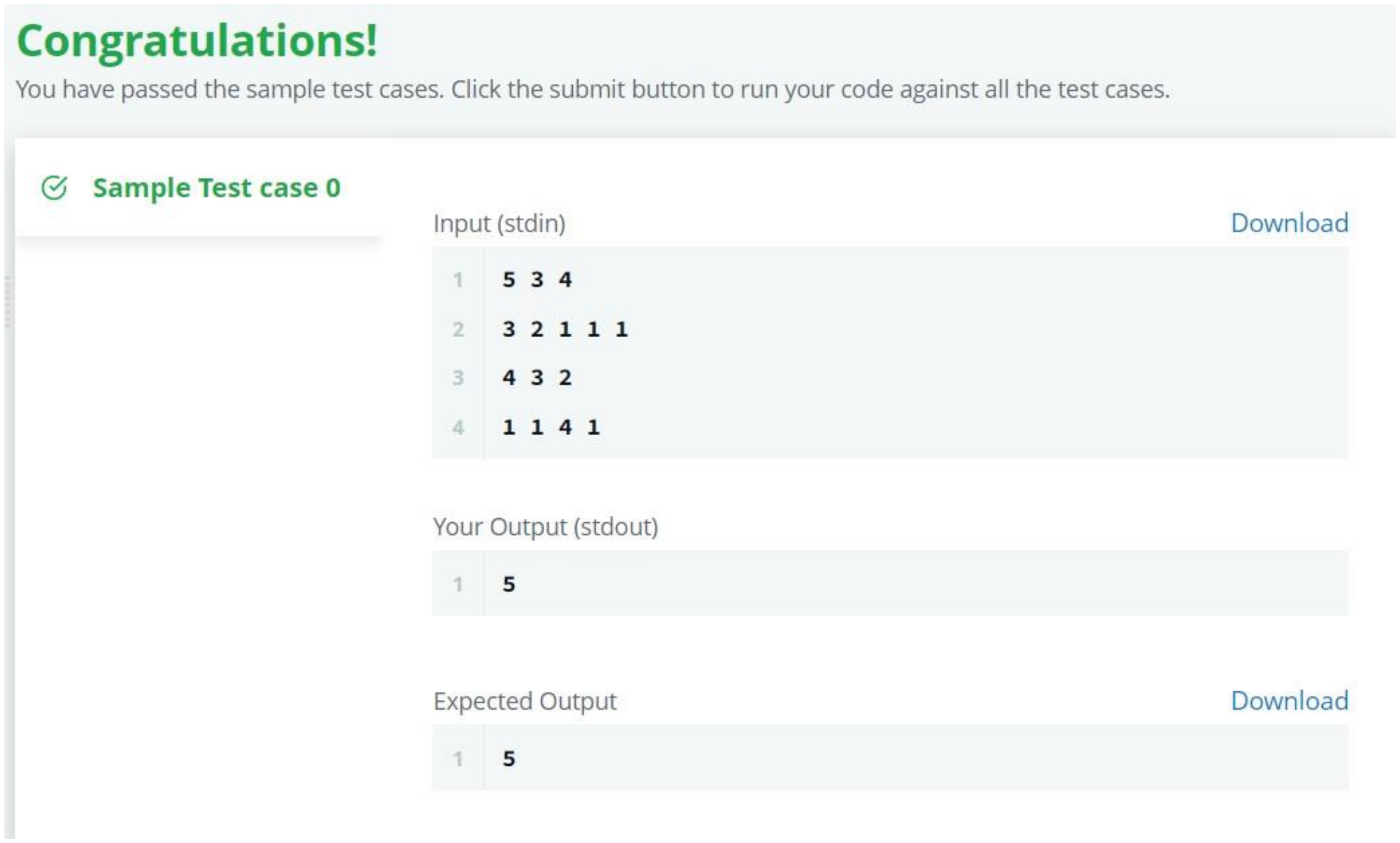
```
int equalStacks(vector<int> h1, vector<int> h2, vector<int> h3) {
int sum1 = accumulate(h1.begin(),h1.end(),0);
    int sum2 = accumulate(begin(h2),end(h2),0);
int sum3 = accumulate(begin(h3),end(h3),0);

    int i = 0, j = 0, k = 0;     while (!(sum1
== sum2 && sum2 == sum3)) {         int temp =
max(sum1, max(sum2, sum3));         if (temp ==
sum1) sum1 -= h1[i++];         else if (temp ==
sum2) sum2 -= h2[j++];         else sum3 -=
h3[k++];
    }
    return sum1;
     }
```

5. Observations/Discussions/ Complexity Analysis:

For this problem this program is done in O(n) complexity where n is the size of largest array.

6. Result/Output/Writing Summary:



Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Input (stdin)

```
5 3 4
3 2 1 1 1
4 3 2
1 1 4 1
```

Your Output (stdout)

```
5
```

Expected Output

```
5
```

EXPERIMENT-2(b)

Aim/Overview of the practical:

To demonstrate the concept of Stack and Queue

2. Task to be done/ Which logistics used: https://www.hackerrank.com/challenges/game-of-two-stacks/problem?isFullScreen=true

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.

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

1. Intialize a variable sum to store the value of sum while iterating through the array and i to iterate through vector a position.
2. iterate through vector a and store the sum of it ith value to sum, till sum <= maxSum and increase i by 1. else if sum> maxSum, break.
3. store the value of i in variable count and create a variable j to iterate through vector b.
4. iterate through vector b while i>0 and j<size of vector b.
5. Add value of jth element of b to sum and increase j.
6. Now if sum>sumMax and i>0, subtract value at ith of vector a from Sum and decrement i till sum .
7. if sum<= maxSum and i+j > count, then count = i+j; 8. Return count.

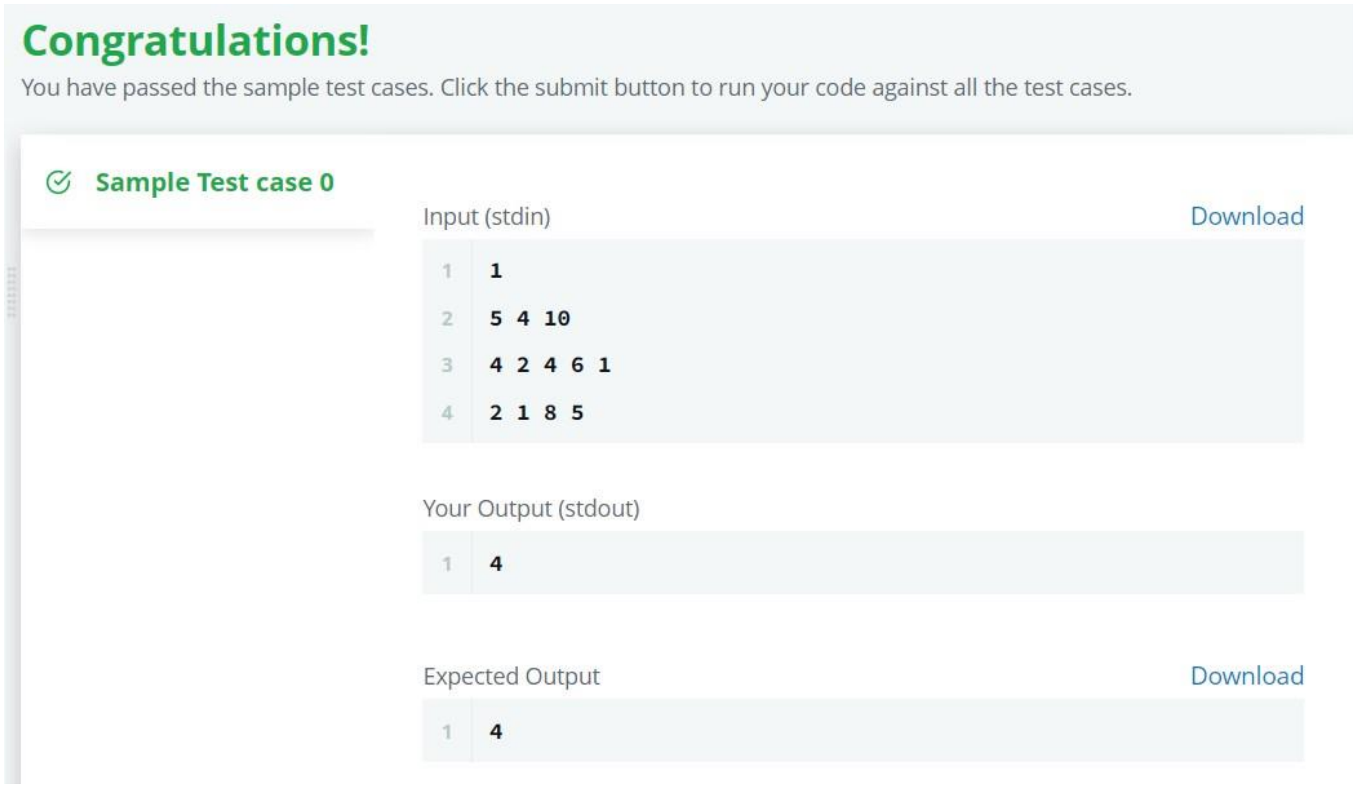
4. Steps for experiment/practical/Code:

```
int twoStacks(int maxSum, vector<int> a, vector<int> b) {
int sum =0,i=0;     for(auto it: a){
if(sum+it<=maxSum) {             sum+=it;
i++;             }
        else break;
    }     int count = i,j=0;
while(j<b.size() && i>=0){
sum+=b[j];         j++;
while(sum>maxSum && i>0){
i--;                 sum-=a[i];
            }             if(sum<=maxSum
&& i+j>count)                count=i+j;
        }
return count;
}
```

5. Observations/Discussions/ Complexity Analysis:

For this problem this code is done in O(nlogn) complexity.

6. Result/Output/Writing Summary:


Congratulations!
You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Input (stdin)
```
1
5 4 10
4 2 4 6 1
2 1 8 5
```
Your Output (stdout)
```
4
```
Expected Output
```
4
```

EXPERIMENT-2(c)

Aim/Overview of the practical:

To demonstrate the concept of Stack and Queue

2. Task to be done/ Which logistics used: https://www.hackerrank.com/challenges/balanced-brackets/problem?isFullScreen=true

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, `]`.

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

- Intialize a stack st using stl. Iterate through string s.
- If s[i] == ‘{‘ or ‘[‘ or ‘(‘, push s[i] in st.
- Else, ○ if(st.empty() || (st.top()!='(' && i==')') || (st.top()!='[' && i==']') || (st.top()!='{' && i =='}')) return **“NO”**
  - ○ Else st.pop()

Now return YES if stack st is empty else NO if not.

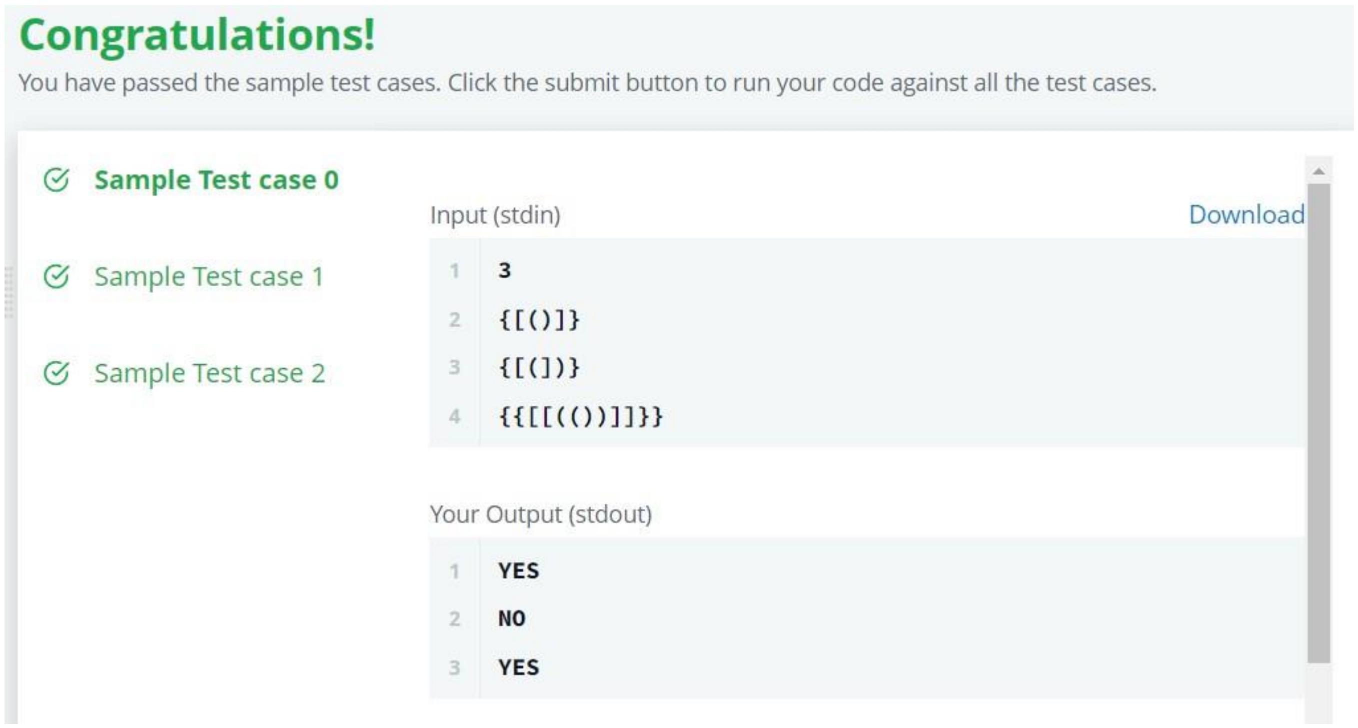
4. Steps for experiment/practical/Code:

```
string isBalanced(string s) {     stack<char> st;
for(auto i: s){          if(i=='{' || i== '[' || i==
'(') st.push(i);          else
             {if(st.empty() || (st.top()!='(' && i==')') || (st.top()!='[' && i==']')
|| (st.top()!='{' && i =='}')){
return "NO";
             }
st.pop();
             }
}
     return st.empty()?"YES":"NO";
}
```

5. Observations/Discussions/ Complexity Analysis:

For this problem this code is done in complexity of O(logn).

6. Result/Output/Writing Summary:



Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Input (stdin)

```
3
{[()]}
{[(])}
{{[[(())]]}}
```

Your Output (stdout)

```
YES
NO
YES
```

EXPERIMENT-2(d)

1. Aim/Overview of the practical:

To demonstrate the concept of Stack and Queue

2. Task to be done/ Which logistics used: https://www.hackerrank.com/challenges/down-to-zero-ii/problem?isFullScreen=true

You are given queries. Each query consists of a single number . You can perform any of the operations on in each move:

1: If we take 2 integers and where , , then we can change

2: Decrease the value of by .

Determine the minimum number of moves required to reduce the value of to.

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

1. Take ll t.
2. Use while loop(t--).
3. Take ll a;
4. Use queue pair with two ll v; 5. Using vector ll with v2(1e6+1,0).
6. Use v.push operation at a,0;
7. If v.size>0 then print front;
8. If z first ==0 return break;
9. Using for loop and if condition print v2.

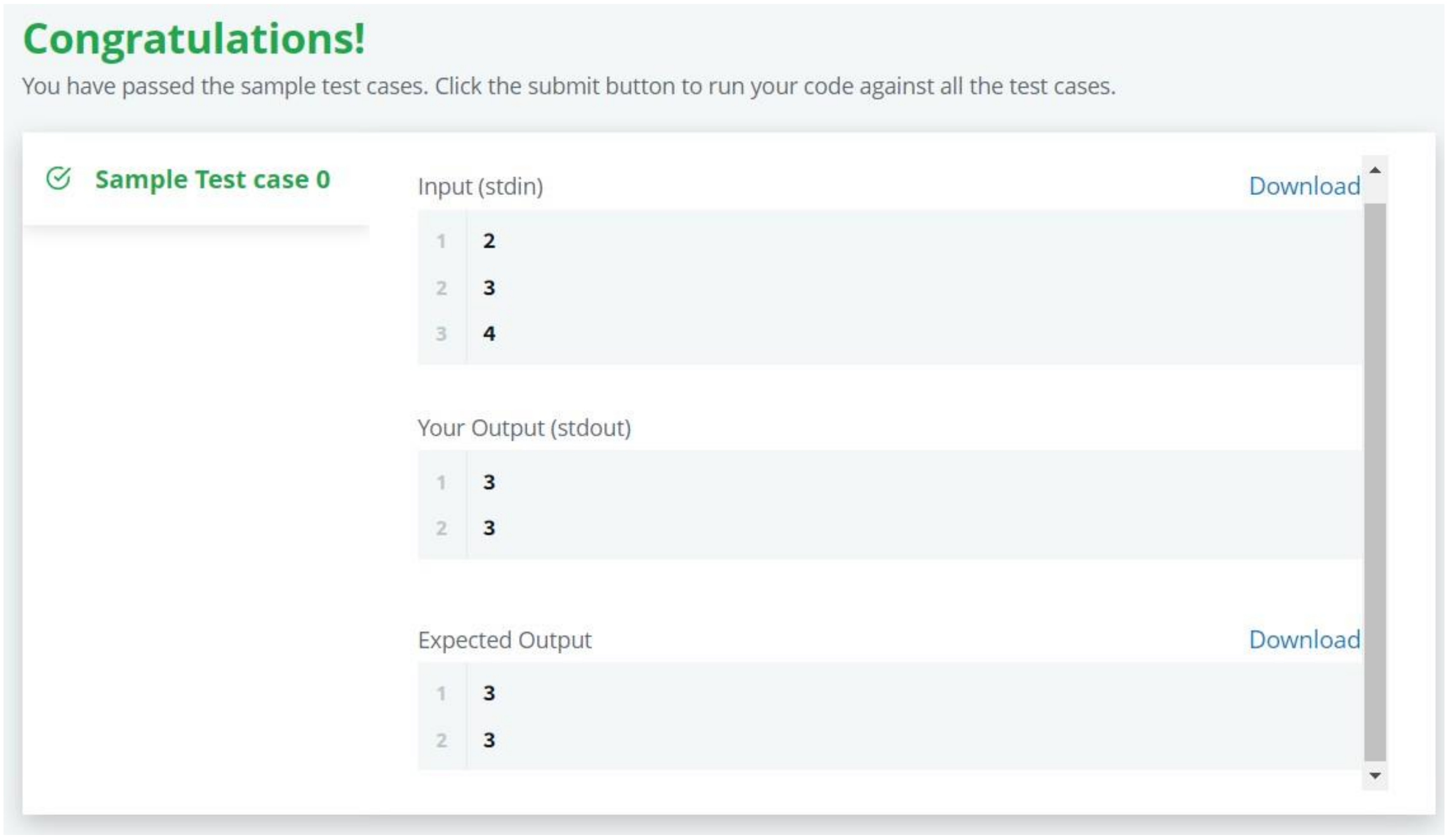
4. Steps for experiment/practical/Code:

```
#include<bits/stdc++.h>
using namespace std;
#define ll long long int
main()
{
    ll t;     cin>>t;
while(t--)     {         ll a;
cin>>a;
queue<pair<ll,ll>>v;
vector<ll>v2(1e6 + 1,0);
v.push({a,0});
while(v.size() > 0)
        {
            pair<ll,ll> z=v.front();
            v.pop();
if(z.first==0)
            {                 break;
}              else{
if(v2[z.first-1]==0)
                {
                    v.push({z.first-1,z.second+1});
v2[z.first-1]=z.second+1;
                }
                for(ll i=sqrt(z.first);i>1;i--)
                {
                    if(z.first%i==0)
                    {
                        if(v2[z.first/i]==0)
                        {
                            v.push({z.first/i,z.second+1});
v2[z.first/i]=z.second+1;
                        }
                    }
                }
            }
}
        cout<<v2[0]<<endl;
    }
}
```

5. Observations/Discussions/ Complexity Analysis:

For this program this code is done in complexity of O(n).

6. Result/Output/Writing Summary:



Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Input (stdin)

```
2
3
4
```

Your Output (stdout)

```
3
3
```

Expected Output

```
3
3
```

EXPERIMENT-2(e)

1. Aim/Overview of the practical:

To demonstrate the concept of Stack and Queue

2. Task to be done/ Which logistics used: https://www.hackerrank.com/challenges/truck-tour/problem?isFullScreen=true

Suppose there is a circle. There are petrol pumps on that circle. Petrol pumps are numbered to (both inclusive). You have two pieces of information corresponding to each of the petrol pump: (1) the amount of petrol that particular petrol pump will give, and (2) the distance from that petrol pump to the next petrol pump.

Initially, you have a tank of infinite capacity carrying no petrol. You can start the tour at any of the petrol pumps. Calculate the first point from where the truck will be able to complete the circle. Consider that the truck will stop at each of the petrol pumps. The truck will move one kilometer for each litre of the petrol.

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

1. Take integer n,p and d both with index size of 100005.
2. Take input of n.
3. Use for loop from i=0 to i<n;
4. Then pre increment i.
5. Take integer ret=amount=sum=0;
6. Repeat step 3.
7. Then p[i] -= d[i];
8. sum += p[i];
9. Use condition if (amount + p[i] < 0)
10. amount = 0;
11. ret = i + 1;
12. else amount += p[i];
13. Print sum >= 0 ? ret : -1

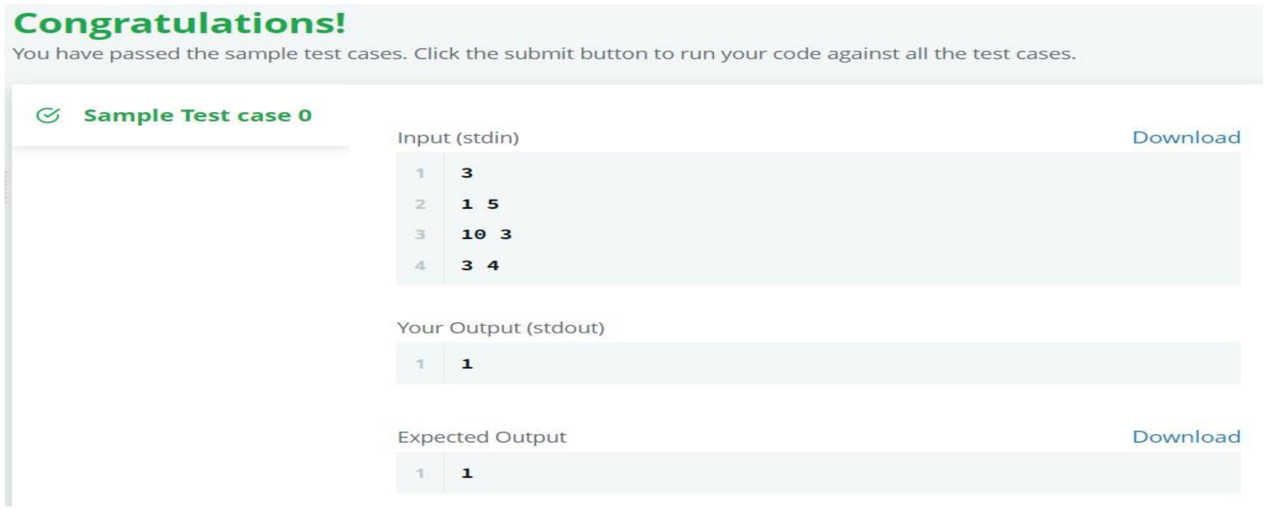
4. Steps for experiment/practical/Code:

```
#include <vector>
#include  <iostream>
#include <algorithm>
using namespace std;
 int n, p[100005], d[100005]; int main() {     scanf("%d",
&n);     for (int i = 0; i < n; ++i) scanf("%d%d", &p[i],
&d[i]);     int ret = 0, amount = 0, sum = 0;     for (int i
= 0; i < n; ++i) {         p[i] -= d[i];         sum +=
p[i];         if (amount + p[i] < 0) {             amount =
0;             ret = i + 1;
        } else amount += p[i];
    }
    printf("%d\n", sum >= 0 ? ret : -1);
return 0;
}
```

5. Observations/Discussions/ Complexity Analysis:

For this program this code is done in complexity of O(n).

6. Result/Output/Writing Summary:


Congratulations!
You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Input (stdin)
```
3
1 5
10 3
3 4
```
Your Output (stdout)
```
1
```
Expected Output
```
1
```

Learning outcomes (What I have learnt):

1. Through this experiment I learn concepts of stacks. **2.**

different operations on stacks: push pop top empty.

3.Learned about traversing and searching in 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.			