**Equilibrium Point: -**

Easy Accuracy: 28.13% Submissions: 477K+ Points: 2

Given an array**A**of **n** positive numbers. The task is to find the first **equilibrium point** in an array. Equilibrium point in an array is a position such that the **sum** of elements **before**it is **equal**to the **sum**of elements **after**it.

**Note:** Return equilibrium point in 1-based indexing. Return -1 if no such point exists.

**Example 1:**

**Input:**

n = 5

A[] = {1,3,5,2,2}

**Output:**3

**Explanation:**

equilibrium point is at position 3 as sum of elements before it (1+3) = sum of elements after it (2+2).

**Example 2:**

**Input:**

n = 1

A[] = {1}

**Output:**1

**Explanation:**

Since there's only element hence its only the equilibrium point.

**Your Task:**  
The task is to complete the function **equilibriumPoint()** which takes the array and n as input parameters and returns the point of equilibrium.

**Expected Time Complexity:**O(n)  
**Expected Auxiliary Space:** O(1)

**Constraints:**  
1 <= n <= 105  
1 <= A[i] <= 109

**Code: -**

//{ Driver Code Starts

#include <iostream>

using namespace std;

// } Driver Code Ends

class Solution{

public:

// Function to find equilibrium point in the array.

// a: input array

// n: size of array

int equilibriumPoint(long long a[], int n){

long long rightsum = 0, leftsum = 0;

for(int i = 0; i<n; ++i)

rightsum += a[i];

for(int i = 0; i<n; ++i){

rightsum -= a[i];

if(leftsum == rightsum)

return i+1;

leftsum += a[i];

}

return -1;

}

};

//{ Driver Code Starts.

int main() {

long long t;

//taking testcases

cin >> t;

while (t--) {

long long n;

//taking input n

cin >> n;

long long a[n];

//adding elements to the array

for (long long i = 0; i < n; i++) {

cin >> a[i];

}

Solution ob;

//calling equilibriumPoint() function

cout << ob.equilibriumPoint(a, n) << endl;

}

return 0;

}

// } Driver Code Ends

**T.C: - O(N)**

**S.C: - O(1)**