First, find the total sum of all the elements of the array. Then, traverse the array. While traversing, find the leftSum for i^th terms and **find rightSum by subtracting leftSum and the next term from total sum.** Compare leftSum with rightSum in each iteration.

Brute Force: Time complexity: $O(n^2)$; Space complexity: $O(1)$

```
int equilibriumPoint(long long a[], int n) // Brute Force
{
    bool flag = false;
    if (n == 1)
        return 1;
    else if (n == 2)
        return -1;

    int sumL = 0;
    int sumR = 0;
    int j;
    for (int i = 0; i < n - 2; i++)
    {
        sumL += a[i];
        j = n - 1;
        while (j - i > 1)
        {
            sumR += a[j];
            j--;
        }
        if (sumL == sumR)
        {
            -----
        }
        sumR = 0;
        if (flag == true)
            break;
    }

    if (flag == true)
        -----
}
```

Optimized: Time complexity: O(n); Space complexity: O(1)

```
for (i = 0; i < n - 1; i++)
    {
        leftSum = leftSum + a[i];
        rightSum = sumTotal - ------;
        if (leftSum == rightSum)
        {
            ----
        }
    }
```