Q1. Given N array elements and Q queries on same array. For each query calculate sum of all elements in given range [L-R]. Note: L ≤ R., L & R are indexes.

arr[10] ⇒ -3 6 2 4 5 2 8 -9 3 1

arr[10]	-3	6	2	4	5	2	8	-9	3	1
	0	1	2	3	4	5	6	7	8	9

Q → 5.

L	R (indexes)	Sum
4	8	9
6	9	3
1	3	12
0	4	14
2	5	13.

Bruteforce

For every query iterate & get sum from [L-R].

```
for ( i=1; i<=Q; i++ ){
      sum=0
    for(j=L; j<=R; j++){
       sum = sum + arr[j]
    }
    print(sum)
}
```

$TC \Rightarrow O(Q * N)$.

$SC \Rightarrow O(1)$

// Given INDIAN team's score for first 10 overs of batting after every order, current score given :-

Overs :-	1	2	3	4	5	6	7	8	9	10
run :-	2	8	14	29	31	49	65	79	88	97

→ total run scored in the last 5 overs :-

$\Rightarrow S[10] - S[5]$

$\Rightarrow 97 - 31$

$\Rightarrow \underline{\underline{66}}$

→ total run scored in the last over :-

$\Rightarrow S[10] - S[9]$

$\Rightarrow 97 - 88 = 9.$

→ total run scored in the 7th over :-

$\Rightarrow S[7] - S[6]$

$\Rightarrow 65 - 49 = 16.$

→ total run scored from over [3 - 6]

$\Rightarrow S[6] - S[2]$

$\Rightarrow 49 - 8 \Rightarrow \underline{\underline{41}}.$

→ total run scored from over [1 - 5]

⇒ S[5] - $\cancel{S[0]}$ → 0

⇒ 31

// cummulative data from start ⇒ prefix sum

pfSum [i] ⇒ sum of all elements from index 0 to i.

pfsum [3] ⇒ sum of all elements from index 0 to 3

pfsum [6] ⇒ sum of all element from idx [0 6]

pfsum [0] ⇒ arr[0]

	0	1	2	3	4	5	6	7	8	9
arr[10] ⇒	-3	6	2	4	5	2	8	-9	3	1
pfsum [] ⇒	-3	3	5	9	14	16	24	15	18	19.

Q

	L	R
idx ⇒	4	8

⇒ pf[8]

⇒ sum [0 8]

⇒ sum [0 3] + sum [4 8] ⇐

pf[8] = sum [0 3] + sum [4 8]

$\Rightarrow$ sum[4 8] = pf[8] − sum[0 3]

= pf[8] − pf[3]

$\Rightarrow$ [3 7] $\Rightarrow$ pf[7]

$\Rightarrow$ sum[0 7]

$\Rightarrow$ sum[0 2] + sum[3 7]

pf[7] = sum[0 2] + sum[3 7]

$\Rightarrow$ sum[3 7] = pf[7] − sum[0 2]

= pf[7] − pf[2]

idea [L R] $\Rightarrow$ pf[R] − pf[L−1]

ea $\Rightarrow$ [0 3] $\Rightarrow$ pf[3] − ~~pf[−1]~~

$\Rightarrow$ pf[3]

```
sum[i j]
    if (i==0)
        ans = pf[j]
    else
        ans = pf[j] - pf[i-1]
```

⇒ Construct Pf sum array :-

$pf[0] = sum[0\ 0] = arr[0]$

$pf[1] = sum[0\ 1] = arr[0] + arr[1]$
$= pf[0] + arr[1]$

$pf[2] = sum[0\ 2] = arr[0] + arr[1] + arr[2]$
$= pf[1] + arr[2]$

$pf[3] = sum[0\ 3] = arr[0] + arr[1] + arr[2] + arr[3]$
$= pf[2] + arr[3]$

$$\boxed{pf[i] = pf[i-1] + arr[i].}$$

```
arr[]    constPrefix( arr , N ) {

             pf[N]  ←  empty array of size N.

           pf[0] = arr[0]
         for( i=1; i<N; i++) {
               pf[i] = pf[i-1] + arr[i]

         }
       return pf[]
}
```

$$\boxed{\begin{array}{l} TC \Rightarrow O(N) \\ SC \Rightarrow O(N) \end{array}}$$

Q1. 2024
Optimised

pf[] = constPrefix(arr, N) → TC ⇒ O(N)

```
for( i = 1; i <= Q ; i++ ) {
    // sum [L R]
    if (L == 0)
        ans = pf[R]
    else
        ans = pf[R] - pf[L-1]
    print(ans)
}
```

⇒ O(Q)

TC :- O(N+Q)
SC :- O(N)

NOTE :- If you are allowed to modify the i/p array, then change the i/p to hold prefix sum, in that case

TC :- O(N+Q)
SC :- O(1)

Q 2. Equillibrium Index :-

Given N array elements, count no. of equillibrium indices.

An index is said to be an equillibrium index if

{ sum of all elements before i^{th} index } = { sum of all elements after i^{th} index }

ex ⇒

	0	1	2	3
arr[4] =	-3	2	4	-1
leftsum =	0	-3	$\boxed{-1}$	3
rightsum =	5	3	$\boxed{-1}$	0

O/p = 1 [index = 2]

Obs :- if $i == 0$, leftsum = 0.

$i == N-1$, rightsum = 0

ex ⇒

	0	1	2	3	4	5	6
arr[7] =	-7	-1	5	2	-4	3	0
leftsum =	0	-7	-8	-3	-1	-5	-2
rightsum =	5	6	1	-1	3	0	0

O/p = 0 [no eq. index].

ex ⇒ arr[] ⇒

	0	1	2	3	4	5	6
arr[]	3	-1	2	-1	1	2	1
leftsum =	0	3	2	4	3	4	6
rightsum =	4	5	3	4	3	1	0

O/P = 2 [indexes - 3, 4]

Bruteforce

For every index, check equilibrium or not, iterate to extreme left & extreme right

```
c=0;
for (i=0; i<N; i++) {
    leftsum = sum[0, i-1]
    rightsum = sum[i+1, N-1]
    if (leftsum == rightsum)
        c++
}
print(c).
```

O(N)

add edge case i==0 & i==N-1

O(N)

O(1) using pfsum array

sum[L R]

↓

O(1) → pfQueries

TC ⇒ O(N²)

SC ⇒ O(1).

→ Optimised using pfsum

TC ⇒ O(N+N) ≈ O(N)

SC ⇒ O(N) → prefix sum.

Q3. Given N array elements, construct a pfEven[] of N,
pfEven[i] = sum of all even indices [0 – i].

		0	1	2	3	4	5
eg ⇒	arr[6] =	3	-2	4	6	-3	5
	pfEven[6] =	3	3	7	7	4	4.

		0	1	2	3	4	5	6	7
eg ⇒	arr[8] =	2	-1	3	1	4	3	2	-1
	pfEven[8] =	2	2	5	5	9	9	11	11

pseudo

```
pfEven[0] = arr[0]
for( i=1; i<N; i++){
    if( i%2 ==1)
        pfEven[i] = pfEven[i-1].
    else
        pfEven[i] = pfEven[i-1]+arr[i]
}
```

$$\boxed{\begin{array}{l} TC \Rightarrow O(N) \\ SC \Rightarrow O(N) \end{array}}$$

Q.4. Given N array elements, and Q queries for each query, calculate sum of all even indices, in given range.

	0	1	2	3	4	5	6	7
arr[8] =	3	4	-2	8	6	2	1	3
pfEven =	3	3	1	1	7	7	8	8

Query

L	R		
2	6	→ ~~4~~ 5	⇒ pfEven[6] − pfEven[1] = 8 − 3 = 5
3	7	→ 7	⇒ pfEven[7] − pfEven[2] = 8 − 1 = 7

\# code similar to Q1, replace pfSum with pfEvenSum.

Sum[i j] → sum of even indexes from i to j.

if (i==0)
    ans = pfEven[j]

else
    ans = pfEven[j] − pfEven[i−1].

$$\boxed{\begin{aligned} TC &\Rightarrow O(N+Q) \\ SC &\Rightarrow O(N) \end{aligned}}$$

~~practice~~

* pfSum[ ]
* pfSumQueries
  sum[L R]
* pfEven[ ]
* pfEvenQueries
* pfOddSum[ ]
* pfOddSumQuries[ ]

↓

sum of odd indx (i j)

	0	1	2	3	~~4~~	5	6	7	8	9
arr[10] =>	-1	3	2	6	~~4~~	2	7	3	2	10
arr[9] =>	-1	3	2	6	2	7	3	2	10	

no impact on LHS

for RHS

new odd idx = old even idx

new even idx = old odd idx

Google

Q5. Spacial Index → HARD.

An index is said to be a spacial index, if after deleting the index, sum of all odd indexes == sum of all the even indexes, print the count of spacial indexes.

eg =>

0	1	2	3	4	5
4	3	2	7	6	-2

Delete index 0

	0	1	2	3	4
arr =>	3	2	7	6	-2

Seven = 8

Sodd = 8

→ index = 0 → special idx.

delete idx 1

	0	1	2	3	4
arr =>	4	2	7	6	-2

Seven = 9

Sodd = 8

X X

delete idx 2

	0	1	2	3	4
arr =	4	3	7	6	-2

Seven = 9

Sodd = 9

→ special index

delete idx 3

	0	1	2	3	4
arr =	4	3	2	6	-2

Seven => 4

Sodd => 9

X X

idx = 4

X X

op = 2

arr[14] $\Rightarrow$ { 0 1 2 3 4 $\boxed{\cancel{5}}$ 6 7 8 9 10 11 12 13 }

to check if 5 is special idx or not,

total sum of all even idx = total sum of all odd idx.

S' $\Rightarrow$ sum after dele
S $\Rightarrow$ sum original

after deletion of 5.

total sum of all even idx after removal $= S'_{Even} = S_{Even}[0\ 4] + S_{odd}[6\ 13]$

$S'_{Even}[6\ 13] = S_{odd}[6\ 13]$

total sum of all odd idx after removal $\Rightarrow S'_{odd} = S_{odd}[0\ 4] + S_{even}[6\ 13]$

$S'_{odd}[6\ 13] = S_{even}[6\ 13]$

if ($S'_{even} == S'_{odd}$)

c++

generalise

$$arr[N] = \underbrace{0\ 1\ 2\ 3\ 4\ .\ i-1}_{\downarrow\ \text{nochange}}\ i\ \underbrace{i+1\ ----\ N-1}_{\downarrow\ \text{change}}$$

$S_{Even} = S'_{even} = [0 \quad i-1]$

$S_{odd} = S'_{odd} = [0 \quad i-1]$

$S'_{Even} = S_{odd}$

$S'_{odd} = S_{Even}$

After removal

$$\boxed{\begin{aligned} T_{Even} &= S_{even}[0 \quad i-1] + S_{odd}[i+1 \quad N-1] \\ T_{odd} &= S_{odd}[0 \quad i-1] + S_{even}[i+1 \quad N-1] \end{aligned}}$$

pseudo

check if all indexes are spacial :-

count = 0.

for(i=0; i<N; i++)

$T_{Even} = S_{even}[0 \quad i-1] + S_{odd}[i+1 \quad N-1]$

$T_{odd} = S_{odd}[0 \quad i-1] + S_{even}[i+1 \quad N-1]$

if($T_{Even} == T_{odd}$)

count++

}

} print(count)

Seven [0 i-1] => PfEven[i-1]

Sodd [0 i-1] => Pfodd [i-1]

Seven [i+1 N-1] => PfEven[N-1] - PfEven[i]

Sodd [i+1 N-1] => PfOdd[N-1] - Pfodd [i]

edge case

i=0

TC => O(N + N + N) ≈ O(N)

SC => O(N + N) ≈ O(N).

0 i-1 $\boxed{}$ i+1 N-1

pfodd → original

pfEven → original

1 2 3 4

→ 4 3 2 3

1 2 $\boxed{3}$

→ 2 3 1

→ 3 4 1 2

→ 2 3 4 1 $\qquad 5 \% 3 = 2$

→ 1 2 3 4

→ 4