Q.1. Given N array elements & Q queries on same array. for each query calculate sum of all elements in given range [L-R]. Note:- L ≤ R.

ex => arr[10] = -3 6 2 4 5 2 8 -9 3 1

-3	6	2	4	5	2	8	-9	3	1
0	1	2	3	4	5	6	7	8	9

Q. 4

L	R → index	
4	8	→ 9
6	9	→ 3
1	3	→ 12
0	4	→ 14.

Brute force

For every query iterate and get sum for [L R].

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

$\boxed{TC \Rightarrow O(Q * N) \quad SC \Rightarrow O(1)}$

// Given Indian team's score, for first 10 overs of batting after every over, current score is given below:

overs :-	1	2	3	4	5	6	7	8	9	10
score till that over :-	2	8	14	29	31	49	65	79	88	97

29 → score after 4 over

97 → score after 10 overs

// total score gained in the last over :- $S[10] - S[9]$
$= 97 - 88$
$= 9.$

// total score gained in the last 5 overs $[6 - 10]$ $= S[10] - S[5]$
$= 97 - 31$
$= 66.$

// total score gained in 7^{th} over $= S[7] - S[6]$
$= 65 - 49 = 16.$

// total score gained from over $[3 \quad 6] = S[6] - S[2]$
$= 49 - 8$
$= 49.$

// total score gained from over $[1 \quad 5] = S[5] - 0 = 31 - 0 = 31.$

* Cummulative data from start ⇒ prefix sum.

pfsum[i] ⇒ sum of all elements from idx 0 to i.

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

pfsum[0] ⇒ sum of all element from idx 0 to 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
pf[10]	⇒	-3	3	5	9	14	16	24	15	18	19.

ex ⇒ sum [4 8] ⇒ pf[8] = sum[0 8]

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

⇒ sum[4 8] = pf[8] − sum[0 3]

= pf[8] − pf[3]

sum[3 7] ⇒ pf[7] = sum[0 7]

$= \underbrace{\text{sum}[0\ 2]} + \text{sum}[3\ 7]$

= pf[2] + sum[3 7]

⇒ sum[3 7] = pf[7] − pf[2].

sum[L R] => pf[R] - pf[L-1].

sum[0 3] => pf[3] - ~~pf[-1]~~ → 0

=> pf[3]

↓ logic

pf[i] → O(1)

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

→ O(1)

sum[L R] → O(1) { given that pfSum[] is cons- }

TC

// Construct pf[]

$pf[0] = sum[0 \; 0] = \underbrace{arr[0]}$

$pf[1] = sum[0 \; 1] = \underbrace{arr[0]}_{pf[0]} + arr[1]$

$\hookrightarrow pf[0] + arr[1]$

$pf[2] = sum[0 \; 2] = \underbrace{arr[0] + arr[1]}_{pf[1]} + arr[2]$

$\hookrightarrow pf[1] + arr[2]$

$pf[3] = sum[0 \; 3] = \underbrace{arr[0] + arr[1] + arr[2]}_{pf[2]} + arr[3]$

$\hookrightarrow pf[2] + arr[3]$

$$\boxed{pf[i] = pf[i-1] + arr[i] \quad (i > 0)}$$

pseudo

```
constPrefix(arr, N) {
    pf[N] → declaring an empty array
            of size N.
    pf[0] = arr[0].
    for( i=1; i<N; i++) {
        pf[i] = pf[i-1] + arr[i]
    }  return pf[];
}
```

TC ⇒ O(N)

SC ⇒ O(N)

arr = [1 2 3 4 5]

pf [1 3 6 10 15

Optimised soln

constPrefix(arr, N); $\longrightarrow O(N)$

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

$\longrightarrow O(Q)$

$TC \Rightarrow O(N+Q)$

$SC \Rightarrow O(N)$

Equilibrium Index

Q2. Given N array elements, count no. of eq. indexes?

An index is an eqbm index, when:-

{ sum of all element before index } = { sum of all elements after index }

eg $\Rightarrow$

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

O/p = 1.

Obs:- if i==0, leftsum = 0 | i==n-1, rightsum = 0.

ex => arr[7] =>

	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

Op = 0

0 ... i-1 | i | i+1 ... N-1

ex =>

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

Op = 2

Bruteforce

for every index check if equ index ?

```
ans = 0
for( i = 0; i < N; i++ ) {  → O(N)
    leftSum = sum(0, i-1)       → loop    ] 0 i-1     ] O(N)
    rightsum = sum(i+1, N-1)    → loop    ] i+1 N-1
    if( leftSum == rightSum ) {
            ans++
    }
}      print(ans)
```

TC => $O(N^2)$

SC => $O(1)$

sum(0 i-1) $\Rightarrow$ pf[i-1]

sum(i+1 N-1) $\Rightarrow$ pf[N-1] - pf[i]

Optimised

```
pf = constPrefix(arr, N)
for(i=0; i<N; i++) {
    leftSum = pf[i-1]              // sum(0, i-1)
    rightsum = pf[N-1] - pf[i]     // sum(i+1, N-1)
    if( leftSum == rightSum) {
         ans++
    }
}   print(ans)
```

$$\boxed{\begin{array}{l} TC \Rightarrow O(N+N) = O(2N) \approx O(N) \\ SC \Rightarrow O(N) \end{array}}$$

Q3. <u>Pf Even:</u>

Given N array elements, construct a pfEven[] of N,

pfEven[i] $\Rightarrow$ sum of all even indexes till i.

		0	1	2	3	4	5
arr[6]	$\Rightarrow$	3	-2	4	6	-3	5
pfEven[]	$\Rightarrow$	3	3	7	7	4	4.

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

<u>pseudo</u>

```
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]
    }
}
```

Q.4. Given N array elements, Q queries. for each query, Calculate sum of all even indexes in given [L-R] 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.

sum even [2 6] ⇒ pfEven[6] − pfEven[1]

⇒ 8 − 3 = 5.

$sum_{even}[3\ 7] = pfEven[7] - pfEven[2]$

$= 8 - 1 \Rightarrow 7.$

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

TC => O(Q+N)

SC => O(N)

* pf[ ]
* pfQueries[ ] → sum(L R) all indexes.
* pfEven[ ]
* pfEvenQueries[ ] → sum(L R) even indexes.
* pfOdd[ ]
* pfOddQueries[ ] → sum(L R) odd indexes

How

$\boxed{9:05 \ / \ 8:35}$

Hint

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

2 7 3 2 10

-1 3 2 6 → no change

2 7 3 2 10 → all odd → even index

all even → odd idx.

Special Index → Google → HARD.

Q.S. An index is said to be a special index, if after deleting it, sum of all even idx = sum of all odd ind.

print the count of special idx.

ex ⇒

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

delete idx 0

	0	1	2	3	4
arr ⇒	3	2	7	6	-2

Sodd = 8

Seven = 8.

idx = 0 ✓

delete idx 1

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

Sodd = 8

Seven = 9

X X

$$\begin{array}{lccccc} & 0 & 1 & 2 & 3 & 4 \\ \text{arr} = & 4 & 3 & 7 & 6 & -2 \end{array}$$

delete idx 2

Sodd = 9
Seven = 9] ✓

delete idx => 3 X X

delete idx => 4 X X

op = 2

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

SI if: total sum of even indexes = total sum of odd indexes

after deletion

i_{Even} = Seven[0 4] + Sodd[6 13]

newSumeven[6 13] = old Sodd[6 13]

$T_{odd} = S_{odd}[0\ 4]$

$+ S_{even}[6\ 13].$

new S_{even} odd $[6\ 13]$

$=$ old $S_{even}[6\ 13]$

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

ans++

generalise

$$\text{arr}[N] = \underbrace{0\ 1\ 2\ 3\ 4 \ldots\ i-1}_{\text{no change}}\ \cancel{i}\ \underbrace{i+1 \ldots\ N-1}_{\text{shift}}$$

(odd → even
even → odd)

sum of even idx after removing i

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

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

pseudo

```
cons ⟶ pfEvenSum, pfOddSum
ans = 0
for( i = 0; i < N; i++) {

    Todd = Sodd [0 i-1] + Seven [i+1 N-1]

    Teven = Seven [0 i-1] + Sodd [i+1 N-1].

    if( Todd == Teven)
        ans++
}

return ans;
}.
```

TC => O(N + N + N)
≈ O(N)

SC => O(N + N)
≈ O(N)

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].

if (i == 0)

pfEven [i-1] => ~~pfEven [-1]~~ → 0

	1	2	3	4	5	6	7
	100	150	80	90	50	100	200
cum =>	100	250	330	420	470	570	770

770 - 420

= 350

			i		
arr =>	2	3	6	-2	0

	0	1	2	3
arr =>	2	3	-2	0

$$\text{Sum}_{\text{Even}} = 2 + (-2) = 0$$

$$\text{Sum}_{\text{odd}} = 3 + 0 = 3$$

X

$$[2^{31}-1,\ 2^{31}-1,\ 2^{31}-1]$$

$$\rightarrow [2^{31}-1,\ \underline{2(2^{31}-1)},\ 3(2^{31}-1)]$$

$\uparrow$

int