Q → Given an array with daily profit / loss from a particular stock. Calculate the total profit / loss over a given range of days.
The function should be able to handle multiple queries for different ranges.

0	1	2	3	4	5	6
-5	10	20	40	50	-10	8

A = [-5 10 20 40 50 -10 8]

Queries

Start	End	
1	4	→ 120
2	5	→ 100
0	0	→ -5

Given an integer array with N elements & Q queries, for every find sum of elements from index L to R.

0	1	2	3	4	5	6
-5	10	20	40	50	-10	8

A = [-5 10 20 40 50 -10 8]

Queries

L	R	
1	4	→ 120
2	5	→ 100
0	0	→ -5

Range Sum Queries

Bruteforce → ∀ Queries, iterate from index L to R.

TC = O(N * Q)

SC = O(1)

```
for i → 0 to (Q-1) {
    // L, R
    for i → L to R {
```

Sum += A[i]

}

}

Cricket

Over →	0	1	2	3	4	5	6	7	8	9	10
Score →		2	8	14	29	31	49	65	79	88	97

Runs scored in 7^{th} over → 65 − 49 = 16

Runs scored from 6^{th} to 10^{th} over → 97 − 31 = 66

Runs scored in 10^{th} over → 97 − 88 = 9

Runs scored from 3^{rd} to 6^{th} over → 49 − 8 = 41

Runs scored from 4^{th} to 9^{th} over → 88 − 14 = 74

Prefix Sum → $P[i] = A[0] + A[1] + \ldots + A[i]$

$P[2] = A[0] + A[1] + A[2]$

$P[3] = A[0] + A[1] + A[2] + A[3]$

$= P[2] + A[3]$

$P[i] = P[i-1] + A[i]$

$P[0] = A[0]$

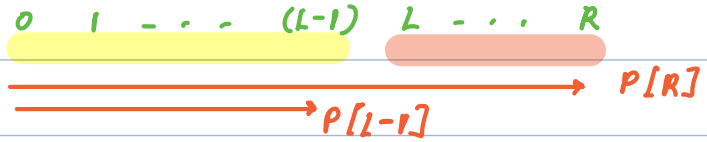
```
P[0] = A[0]
for i → 1 to (N-1) {
    P[i] = P[i-1] + A[i]
}
```

TC = O(N)

	0	1	2	3	4	5
A = [	10	32	6	12	20	1]
✓ P → [	10	42	48	60	80	81]

Sum (L to R)

P[R] − P[L−1]



sum (0 — 3) → P[3] - P[0-1]

index out of bound error

```
for i → 0 to (Q-1) {
    // L , R
    if (L == 0)  print (P[R])
    else  print (P[R] - P[L-1])
}
```

TC = O(Q)

TC = O(N + Q) SC = O(N)

O(1) → Use A[] to store prefix sum.

~~A[0] = A[0]~~

```
for i → 1 to (N-1) {
    A[i] = A[i-1] + A[i]
}
```

	0	1	2	3	4	5
A = [	10	~~32~~ 42	~~6~~ 48	~~12~~ 60	~~20~~ 80	~~1~~ 81]

A[i] = P[i] - P[i-1]

Q → Given an integer array & Q queries (L, R)
Find the sum of all even indexed elements.

	0	1	2	3	4	5
A = [	2	3	1	6	4	5]

Queries

1 3 → 1

2 5 → 5

3 3 → 0

Bruteforce → TC = O(N * Q)

SC = O(1)

P[i] → A[0] + A[2] + A[4] . . .

	0	1	2	3	4	5
A = [	2	3	1	6	4	5]
P →	2	2	3	3	7	7

	0	1	2	3	4
A = [	2	4	3	1	5]
P →	2	2	5	5	10

```
P[0] = A[0]
for i → 1 to (N-1) {
    if (i % 2 == 0)    P[i] = P[i-1] + A[i]
    else               P[i] = P[i-1]
}
```

```
for i → 0 to (Q-1) {  // Queries → L = query[i][0]
    // L, R                       R = query[i][1]
    if (L == 0)  print (P[R])
    else         print (P[R] - P[L-1])
}
```

TC = O(N + Q) SC = O(N) → O(1)

Find the sum of all odd indexed elements.

```
P[0] = 0
for i → 1 to (N-1) {
    if (i % 2 == 1)    P[i] = P[i-1] + A[i]
    else               P[i] = P[i-1]
}
```

Q → Given an integer array, count the no. of special index i.e. index after removing which sum of all even indexed elements = sum of all odd indexed elements.

i	A = [3 (0)	2 (1)	7 (2)	6 (3)	-2 (4)]	Seven	Sodd
0 →	2	7	6	-2		8	≠ 5
1 →	3	7	6	-2		9	≠ 5
2 →	3	2	6	-2		9	≠ 0
3 →	3	2	7	-2		10	≠ 0
4 →	3	2	7	6		10	≠ 8

Ans = 0

A = [4 (0) 1 (1) ~~3~~ (2) 7 (3) 10 (4)]

7 10

odd sum = 1 + 10 = 11

A = [2 (0) 3 (1) 1 (2) ~~4~~ (3) 0 (4) -1 (5) 2 (6) -2 (7) 10 (8) 8 (9)]

0 -1 2 -2 10 8

odd sum = 3 + 0 + 2 + 10 = 15

∀ elements > i^{th} index → odd index move to even
even index move to odd

A = [2 (0) 3 (1) 1 (2) ~~4~~ (3) 0 (4) -1 (5) 2 (6) -2 (7) 10 (8) 8 (9)]

0 -1 2 -2 10 8

even sum = 2 + 1 + (-1) + (-2) + 8 = 8

check i^{th} index

Seven = S even (0 — i-1) + S odd (i+1 — N-1)

Sodd = S odd (0 — i-1) + S even (i+1 — N-1)

$Sum_{even} = evenP[i-1] + (oddP[N-1] - oddP[i])$

$Sum_{odd} = oddP[i-1] + (evenP[N-1] - evenP[i])$

```
// calculate oddP[] & evenP[]
cnt = 0
for i → 0 to (N-1) {
    if (i == 0) {
        se = oddP[N-1] - oddP[i]
        so = evenP[N-1] - evenP[i]
    } else {
        se = evenP[i-1] + (oddP[N-1] - oddP[i])
        so = oddP[i-1] + (evenP[N-1] - evenP[i])
    }
    if (se == so) cnt++
}
return cnt
```

TC = O(N) SC = O(N)