Q.1 Given an Array. Create and return prefix sum Array.

where PS[i] = A[0] + A[1] + A[2] + ... + A[i].

A = [2 4 5 -3 17 8]
    0 1 2 3 4 5

PS = [2 6 11 8 25 33]
     0 1 2 3 4 5

A = [1, 2, 0, 4, -3, 5]
    0 1 2 3 4 5

PS = [1 3 3 7 4 9]
     0 1 2 3 4 5

1st approach

→ for every index i, calculate the sum from 0 to i

TC: $O(n^2)$

optimized version

A = [1, 2, 0, 4, -3, 5]
0 1 2 3 4 5

ps = [1 3 3 7 4 9]
0 1 2 3 4 5

ps[0] = A[0]

ps[1] = A[0] + A[1] ⇒ ps[0] + A[1]
ps[0]

ps[2] = A[0] + A[1] + A[2] ⇒ ps[1] + A[2]
ps[1]

ps[3] = A[0] + A[1] + A[2] + A[3] ⇒ ps[2] + A[3]
ps[2]

ps[i] = ps[i-1] + A[i]

A = [1, 2, 0, 4, -3, 5]
0 1 2 3 4 5

ps = [1 3 3 7 4 9]
0 1 2 3 4 5

```
int[]  prefixSum (int[] A) {
    int n= A.length;
    int[] ps= new int[n];
    ps[0]= A[0];
    for(int i=1; i<n; i++) {
        ps[i]= ps[i-1] + A[i];
    }
    return ps;
}
```

TC: O(n)

Q.2 Range Sum queries

Given an array and Q queries. Find the answer for all queries in the given range.

$1 \le n \le 10^5$

$1 \le Q \le 10^5$

A = [3 4 -2 6 8 10 13 1]
0 1 2 3 4 5 6 7

Q = 4 (L <= R)

L	R	ans
1	3	8
2	6	35
5	5	10
0	3	11

```
void solve (int [] A, int [][] Q) {

    for(int i=0; i < Q.length; i++) {
        int l = Q[i][0];
        int r = Q[i][1];
        //find sum of A[] from l to r
        int sum = 0;
        for (int k=l; k<=r; k++) {
            sum += A[k];
        }
        sop(sum);
    }
}
```

A = [2 9 4 5]
0 1 2 3

Q = [[0,2]
[1,3]
]

i=0, l=0 r=2, Sum=15

i=1, l=1 r=3, Sum=18

O(Q*N) → TLE

Improvised idea

A = [3 4 -2 6 8 10 13 1]
0 1 2 3 4 5 6 7

ps = [3 7 5 11 19 29 42 43]
0 1 2 3 4 5 6 7

sum(2, 5) = sum(0, 5) − sum(0, 1)

ps[5] − ps[1] = 29 − 7 = 22

sum(3, 6) = sum(0, 6) − sum(0, 2)

ps[6] − ps[2] = 42 − 5 = 37

sum(0, 4) = ps[4]

***

sum(L to R) = ps[R] − ps[L−1]

(L != 0)

go on query → L, R

```
if (L == 0) {
        sop (PS[R]);
}
else {
        sop( PS[R] - PS[L-1]);
}
```

```
void solve (int[] A, int[][] Q) {
    int[] PS = prefixSum(A);

    for(int i=0; i < Q.length; i++) {
        int L = Q[i][0];
        int R = Q[i][1];

        //find sum of A[] from L to R
        if (L == 0) {
            sop (PS[R]);
        }
        else {
            sop (PS[R] - PS[L-1]);
        }
    }
}
```

TC: O(N + Q)

Q.3 Equilibrium Index

Amazon, Adobe

Given an Array, find the equilibrium index.

i is an equilibrium index when :

sum of all elements on the left side of i = sum of all elements on the right side of i

A: [-7 5 1 2 -4 3 0]
0 1 2 3 4 5 6

ans = 3

A: [5 1 3 2 9]
0 1 2 3 4

ans = 3

A: [1 2 3]
0 1 2

ans = -1

brute force → hw

for every index i
→ find ls
→ find rs
and compare ls and rs

Sum of all elements on the left side of i	=	Sum of all elements on the right side of i
sum(0, i-1)		sum(i+1, n-1)
ps[i-1]		ps[n-1] - ps[i]

```
int equilibriumIndex(int[] A) {
    int[] ps = prefixSum(A);
    for (int i=0; i< A.length; i++) {
        int ls = 0;
        if (i > 0) {
            ls = ps[i-1];
        }
        int rs = ps[n-1] - ps[i];
        if(ls == rs) {
            return i;
        }
    }
    return -1;
}
```

TC: O(N)

SC: O(N)

A: [5 1 3 2 9]
0 1 2 3 4

ps: [5 6 9 11 20]
0 1 2 3 4

i	ls	rs
0	0	20-5 = 15
1	5	20-6 = 14
2	6	20-9 = 11
3	9	20-11 = 9

Q.4 Given an Array and Q queries, find the count of even numbers for every query.

A :	[3	5	8	9	16	14	13	12]
	0	1	2	3	4	5	6	7

Queries

L	R	ans
1	5	3
2	6	3
4	5	2
4	4	1
3	6	2

A :	[3	5	8	9	16	14	13	12]
	0	1	2	3	4	5	6	7

pc :	[0	0	1	1	2	3	3	4]
	0	1	2	3	4	5	6	7

pc[i]
↳ no. of even elements in A[] from 0 to i.

A : [3 5 8 9 16 14 13 12]
0 1 2 3 4 5 6 7

pc : [0 0 1 1 2 3 3 4]
0 1 2 3 4 5 6 7

1, 5 ⟹ pc[5] - pc[0] = 3 - 0 = 3

3, 7 ⟹ pc[7] - pc[2] = 4 - 1 = 3

```
void solve (int[] A, int[][] Q) {
    int[] pc = prefixfuncount(A);

    for(int i=0; i<Q.length; i++) {
        int L = Q[i][0];
        int R = Q[i][1];
        if(L==0) {
            sop(pc[R]);
        }
        else {
            sop(pc[R] - pc[L-1]);
        }
    }
}
```

TC: O(N+Q)

A : [3 5 8 9 16 14 13 12]
0 1 2 3 4 5 6 7

pc : [0 0 1 1 2 3 3 4]
0 1 2 3 4 5 6 7

```
int[] prefixEvenCount(int[] A) {
    int n = A.length;
    int[] pc = new int[n];

    if(A[0] % 2 == 0) {
        pc[0] = 1;
    }
    else {
        pc[0] = 0;
    }

    for(int i=1; i<n; i++) {
        int temp=0;
        if(A[i] % 2 == 0) {
            temp = 1;
        }
        pc[i] = pc[i-1] + temp;
    }

    return pc;
}
```

A = [2 9 4 3 16]
0 1 2 3 4

pc = [1 1 2 2 3]
0 1 2 3 4

Doubts

2^N, 3^N → exponentials

$x^5 + x^4 + 3x^2$ → polynomial

$x^2 + 3x + 2$ → quadratic (power 2)

buy and sell stock:

19 24 5 16 72

track of left_min

pick from both sides

B = 4

A = [2 9 5 6 7 -3 4]

possibilities:

pick from left	pick from right
4	0
3	1
2	2
1	3
0	4

```
      i       j
A = [ 2  9  5  1  7  -3  4 ]      B = 4
      0  1  2  3  4  5  6
```

sum = 2+9+5+1 , ans = 17

i → B-1
j → n-1

i	j	Sum
3	6	2+9+5+1-1+4
2	5	2+9+5+4-5+(-3)
1	4	2+9+4+(-3)-9+7
0	3	2+4+(-3)+7-2+1

```
i >= 0 {
    sum- = A[i]
    sum+ = A[j]
    i--;
    j--;
}
```

```
for (int i=0; i< 2^n ; i++) {
    int j=i;
    while (j>0) {
        j= j-1;
    }
}
```

i	j [i 1]	itr
0	0 to 1	0
1	1 to 1	+ 1
2	2 to 1	+ 2
3	3 to 1	+ 3
⋮		+ ⋮
2^n-1	2^n-1 to 1	$2^{n-1}-1$

$0+1+2+3+\ldots\ldots+(2^{n-1}-1)$

Sum of $2^{n-1}-1$ natural no.

Sum of N : $\frac{N(N+1)}{2}$

$$\Rightarrow \frac{(2^{n-1}-1)(2^{n-1}-x+x)}{2}$$

$$\Rightarrow \frac{2^{2n-2}-2^{n-1}}{2}$$

$2^{2n} \Rightarrow (2^2)^n$

$2^{2n} \approx 4^n$

```
for(int i=n; i>=1; i=i/2) {
    for(int j=1; j<i; j++) {
        sop( );
    }
}
```

i	j	itr
n	$[1 \quad n)$	$n-1$
$\frac{n}{2}$	$[1 \quad \frac{n}{2})$	$\frac{n}{2}-1$
$\frac{n}{4}$	$[1 \quad \frac{n}{4})$	$\frac{n}{4}-1$
$\vdots$		$\vdots$
1	$[1 \quad 1)$	0

$$n-1+\frac{n}{2}-1+\frac{n}{4}-1+\cdots+0$$

$$n+\frac{n}{2}+\frac{n}{4}+\cdots+0-(1*t)$$

$$a=n$$

$$r=\frac{1}{2}$$

$$t=\log_2 n$$

TC: $O(N)$

$=$

```
for(int i=1; i<=n; i=i*2) {
    for(int j=1; j<=n; j++) {
        SOP( );
    }
}
```

i	j	itr
1	1 → N	N
2	1 → N	+ N
4	1 → N	+ N
⋮		+ N + ...
N	1 → N	N

$N * \log_2 N$

```
int i=1;
while (i<n) {
    int x=i;
    while (x-- > 0) {
    }
}
```

i	x	itr
1	1 → 1	1
2	2 → 1	+ 2
3	3 → 1	+ 3
⋮		+ ...
n-1	n-1 → 1	n-1

$1+2+3+..+n-1$

$\frac{N(N+1)}{2}$ $\Rightarrow$ $N \to n-1$

$\frac{(n-1)(n)}{2}$

TC: $O(n^2)$