Q1) Given N elements, print max subarray sum of len = k.

Ex1 arr[8] = { -3 2 6 4 1 -4 5 3 } (indices 0 1 2 3 4 5 6 7)

k = 4

s	e	sum
0	3	9
1	4	13
2	5	7
3	6	6
4	7	5

$\boxed{ans = 13}$

TC ⇒ no of subarrays × k

$e = [k-1, n-1]$

$n \cancel{-1} - (k-1) \cancel{+1}$

$\Rightarrow n-k+1$

no of Subarrays $= n-k+1$

Pseudo Code !

```
int s = 0, e = k-1;
max_val = Integer.Min;
while ( e<n ) {
    // iterate from s to e to
       get sum.
    int sum = 0;
    for (int i=s ; i<=e ; i++) {
        sum = sum + arr[i];
    }
    max.val => max(max.val, sum);
    s++, e++;
}
print max_val;
```

↓

$Tc : (n-k+1)(k)$

$k = 1$

$Tc = O(n)$

$k = n/2$

$Tc : O(n^2)$

$SC : O(1)$

$k = N$

$Tc = O(n)$

Approach 2 :

1) Use Prefix sum.

```
if (s == 0)
    sum = pf[e];
else
    sum = pf[e] - pf[s-1]
```

$Tc : O(n)$, $SC : O(N)$

$Tc =$ no of subarrays

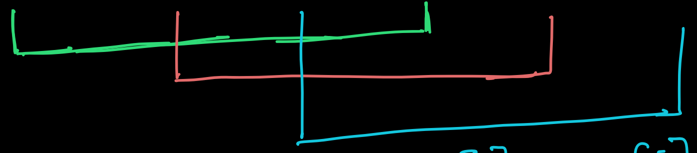
$= (n-k+1)$

$Tc : O(n)$

worst case

Approach 3:

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



$- arr[1] + arr[5]$

s	e	sum	
0	3	9	⇒ loop & find sum.
1	4	13	⇒ $sum - arr[0] + arr[4]$ $9 - (-3) + 1$ ⇒ 13
2	5	7	⇒ $sum - arr[1] + arr[5]$ ⇒ $13 - 2 + (-4)$ ⇒ 7
3	6	6	⇒ Similar track
4	7	5	

for s & e : $sum - arr[s-1]$
$+ arr[e]$

[carry forward + subarrays of same length = SLIDING WINDOWS]

Pseudo Code

```
int s = 0 , e = k-1;
max_val = Integr.Min;
Sum = 0
for (int i = s ; i ≤ e ; i++) {          // first window
                                          k iterations
      Sum = sum + arr[i];
}
max_val = sum;
S = 1 , e = k;
while (e < n) {

      Sum = sum - arr[s-1] + arr[e];
      max_val = max (max_val, sum);
      s++; e++;

}

return max_Val;
```

→ $[k, n-1]$

$n \cancel{-1} - k \cancel{+1}$

$= n-k$ iterations.

TC : O(n)

SC : O(1)

Q2 Given arr[] and a number B. Find and return minimum no of swaps to bring all numbers $\le B$ together.

Ex1 arr = { 1 12 10 3 14 10 5 }, $B = 8$

ans = 2

Ex2 arr[] = { 19 11 3 9 7 25 6 20 4 }, $B = 10$

ans = 1

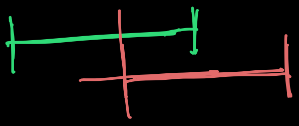
1) Find the count of elements $\le B$. Let's say it is K. This is the window size

2) good elements ⇒ elements $\le B$.

3) For a subarray of len K. If number good element = X

Swaps ⇒ $K - X$

{ 1 12 10 3 14 10 5 } , B = 8



1st window : no of good elements (ng) ⇒ 1

```
2nd window :  if (arr [s-1] ≤ B)
                  ng--;
              if (arr [e] ≤ B)
                  ng++;
```

Pseudo Code!

```
1) // find size of window.
   for (int i=0 ; i<n ; i++) {
           if (arr [i] ≤ B)
               K++;
   }
```

// K is the size of window.

```
2) // Edge Cases.
      if (k==0 || k=1 || k==n)
                return 0;

3) // first window
     ng=0
    for (int i=0 ; i< k-1 ; i++) {
          if (arr[i] ≤ B)
                 ng++;
    }
     min_val => k-ng;

4) // Sliding window.
     int s = 1 , int e = k;
     while (e<n) {
            if (arr[s-1] ≤ B)
                    ng--;
            if (arr[e] ≤ B)
                    ng++;
            ans => k-ng;
            min_val => min (ans, min_val);
            s++, e++
   }
```

TC: O(n)

SC: O(1)

0 1 2 3 4 5 6 7 8

arr [] = { 19 11 3 9 7 25 6 20 4 } , B = 10

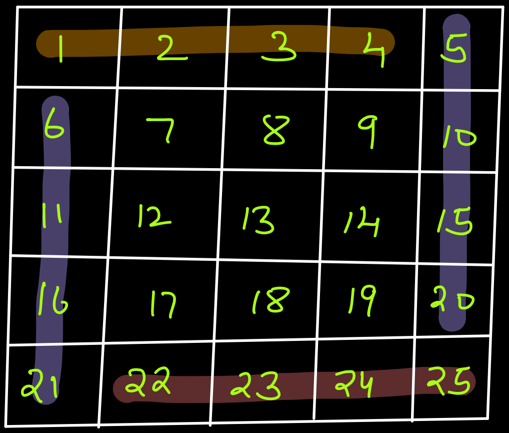
ans = 1 , ~~s => 2, e = 6~~

=> [0 , s-1] => no good elements

[e+1, n-1] => [8]

=> [s, e]

Q3 Given mat[N] [N] , print boundary in clockwise direction.



1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

→ n-1 elements in the first row.

→ n-1 elements in the last column.

→ n-1 elements in the last row. .

→ n-1 elements in the first colum.

TC: $O(m+n)$

SC: $O(1)$

Pseudo Code

$[0, n-1] = n$

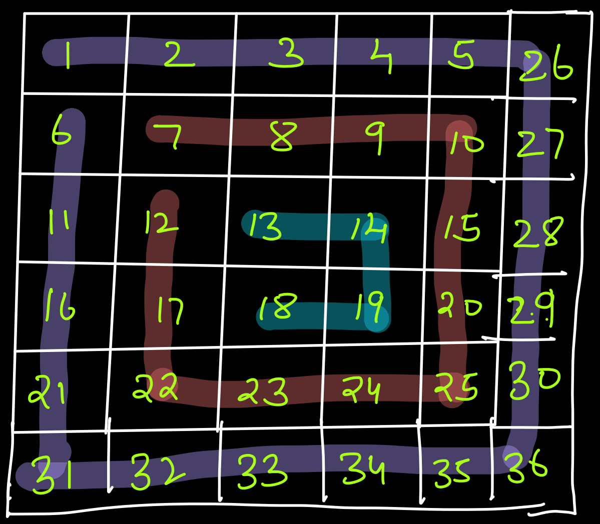
$[1, n-1] = n-1$

```
int i=0, int j=0
for (k=1 ; k<n ; k++) {
        print (mat[i][j]);
        j++;
}                                   ⟹ i = 0, j = n-1
for (k=1 ; k<n ; k++) {
        print (mat[i][j]);
        i++;
}
                                    ⟹ i = n-1, j = n-1

for (k=1 ; k<n ; k++) {
        print (mat[i][j]).
        j-- ;
}                                   ⟹ i = n-1, j = 0


for (k=1 ; k<n ; k++) {
        print (mat[i][j]).
        i-- ;
}
                                    ⟹ i = 0, j = 0
```

Q4 Spiral printing



1	2	3	4	5	26
6	7	8	9	10	27
11	12	13	14	15	28
16	17	18	19	20	29
21	22	23	24	25	30
31	32	33	34	35	36

n ⇒ 6 ⇒ k=5

↓

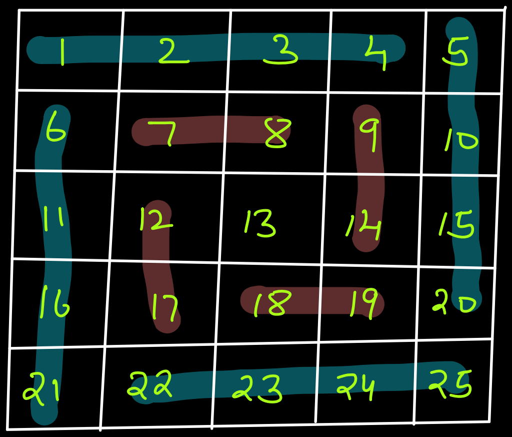
n ⇒ 4 ⇒ k=3

↓

n=2 ⇒ k=1

↓

n=0 ⇒ terminating condition



1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

n=5, k ⇒ 4

n=3, k = 2

n = 1, k = 0

↓

print (mat[i][j])

Pseudo Code

```
int i=0, int j=0
while (n>1) {
        for (k=1 ; k<n ; k++) {
              print (mat[i][j]);
              j++;
        }
        for (k=1 ; k<n ; k++) {
              print (mat[i][j]);
              i++;
        }
        for (k=1 ; k<n ; k++) {
              print (mat[i][j]).
              j--;
        }
        for (k=1 ; k<n ; k++) {
              print (mat[i][j]).
              i--;
        }
        n = n-2; i++; j++;
}
    if (n==1)
          print (mat[i][j]);
}
```

Tc: $O(n^2)$

Sc: $O(1)$