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

length k

0 → k-1

size = 3

[0-2]

[1-3]

[2-4]

[3-5]

⋮

[7-9]

last k length

(N-K → N-1]

$\boxed{0 \rightarrow N-k}$

$(N-K) - 0 + 1$

$= \underline{N - k + 1}$

• arr [N]. Find the max subarray sum of length k.

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

K=5

↓

16

s	e	sum
0	4	7
1	5	8
2	6	12
3	7	16
4	8	10
5	9	11

B.F: → consider all subarray of len k.

```
s=0, e=k-1, ans=INT_MIN
while( e<N )      // s<=N-k
{
    int  sum=0;
    for( int i=s; i<=e; i++)
         sum+=arr[i];
    if(sum>ans)
         ans=sum;
    s++; e++;
}
```

TC: ?

$(N-k+1) * k$

$k=1$ → $O(N)$

$k=N/2$ → $(N-N/2+1)*N/2$

$\approx N^2/4$

$O(N^2)$

$k=N$ → $O(N)$

- using pf sum

```
        // Build pf sum
    s=0, e=k-1, ans=INT_MIN
    while( e<N     )    // s<=N-k
    {
        int  sum=0;
           if(s==0) sum= pf[e];
           else  sum =  pf[e] - pf[s-1];
         if(sum >ans)
            ans= sum;

          s++; e++;
    }
```

(N-K+1)

⇒ $\boxed{\begin{array}{l} T.C: O(N) \\ S.C: O(N) \end{array}}$

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

0 — k-1 $\boxed{0-4}$ Iterate & find sum.

sum = 7

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

$\boxed{1-5}$ sum + arr[5] — arr[0]

sum = 7 + (-2) - (-3)

$\boxed{\text{sum} = 8}$

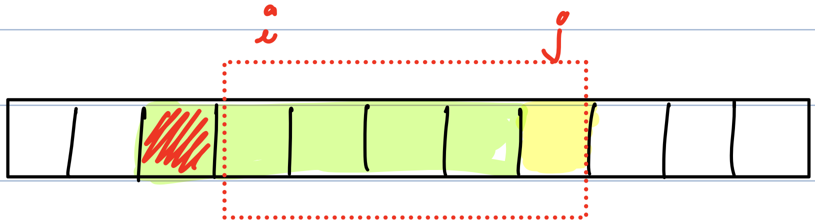
0	1	2	3	4	5	6	7	8	9
-3	4	-2	5	3	-2	8	2	-1	4

$\boxed{2-6}$ sum + arr[6] — arr[1]

4

sum = 8 + 8 - 4

$\boxed{\text{sum} = 12}$



$\boxed{\text{sum}(i-j) = \text{sum} + \text{arr}[j] - \text{arr}[i-1]}$

whenever subarray length is fixed → sliding window

```
// calulate the sum for first window
    ans = INT_MIN , sum = 0;
    for( int i=0; i<K; i++)
            sum += arr[i];
    ans = sum;
```

K

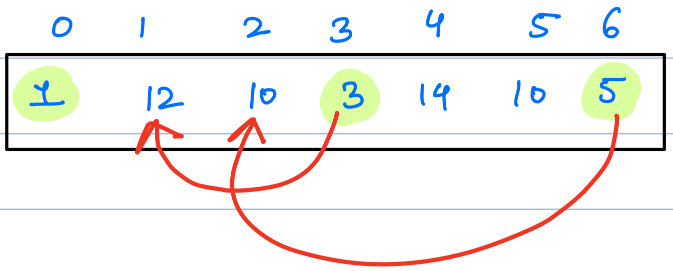
```
// consider remaining windows

      s = 1, e = K
    while( e < N)
    {
          sum = sum + arr[e] - arr[s-1];
          if( sum > ans)
               ans = sum;
          s++; e++;
    }
```

N-K

total = K + N - K
      = N

$\boxed{\begin{array}{l} T.C: O(N) \\ S.C: O(1) \end{array}}$

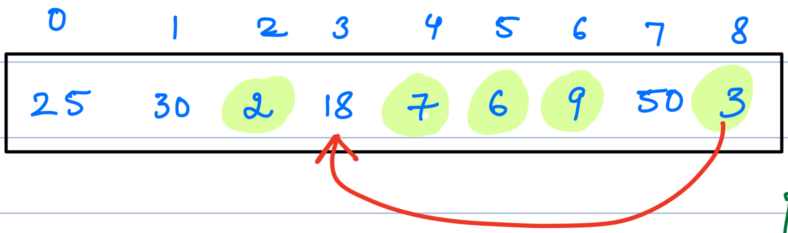
- given arr[N] and a number B. Find and return minimum no of swaps to bring all number <=B together.



0	1	2	3	4	5	6
1	12	10	3	14	10	5

B = 8

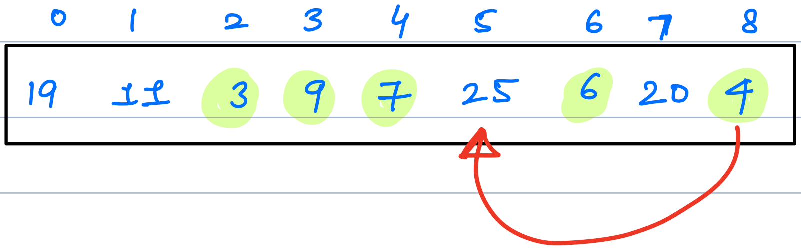
ans = 2



0	1	2	3	4	5	6	7	8
25	30	2	18	7	6	9	50	3

B = 10

ans = 1



0	1	2	3	4	5	6	7	8
19	11	3	9	7	25	6	20	4

B = 10

ans = 1

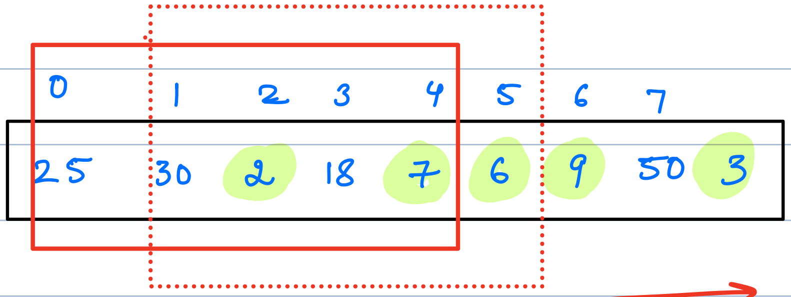
- Bring all elements together → form a subarray
  ↓
  size ?
  ↓
  can we calculate ?
  ↓
  iterate & count

- subarray size is fixed

We'll prefer a subarray of calculated size which have least number of Bad elements

$> B$

B.F:- consider all subarrays
↳ iterate the subarray } → $O(N^2)$



0	1	2	3	4	5	6	7
25	30	2	18	7	6	9	50	3

0 − 4   cnt = 3 → $> B$

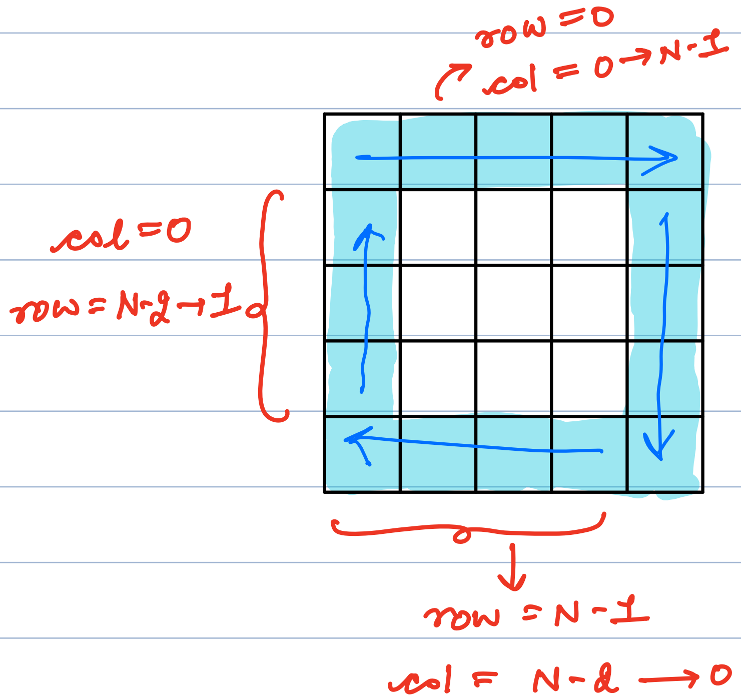
1 − 5

```
if ( arr[e] > B)
    cnt++;
if ( arr[s-1] > B)
    cnt--;
ans = min(ans, cnt);
```

```
// find length of subarray

        k=0;

        for( int i=0; i<N; i++)
            if( arr[i] <=B)
                k++;

                      ↑(>B)
// calulate the cnt for first window
   ans = INT_MIN , cnt =0;
   for( int i=0; i<k; i++)
          if( arr[i] >B) cnt++;
   ans = cnt;                              K

// consider remaining windows

     s=1, e=k
     while( e<N)
K   {
             if( arr[e] >B) cnt++;
             if( arr[s-1] >B) cnt--;

             ans = min( ans, cnt);

             s++; e++;
     }
```

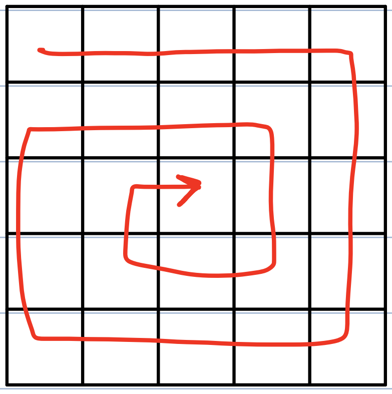
10: 40 pm

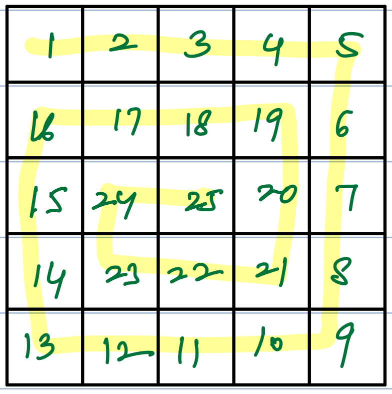
matrix [N][N] → print the boundary



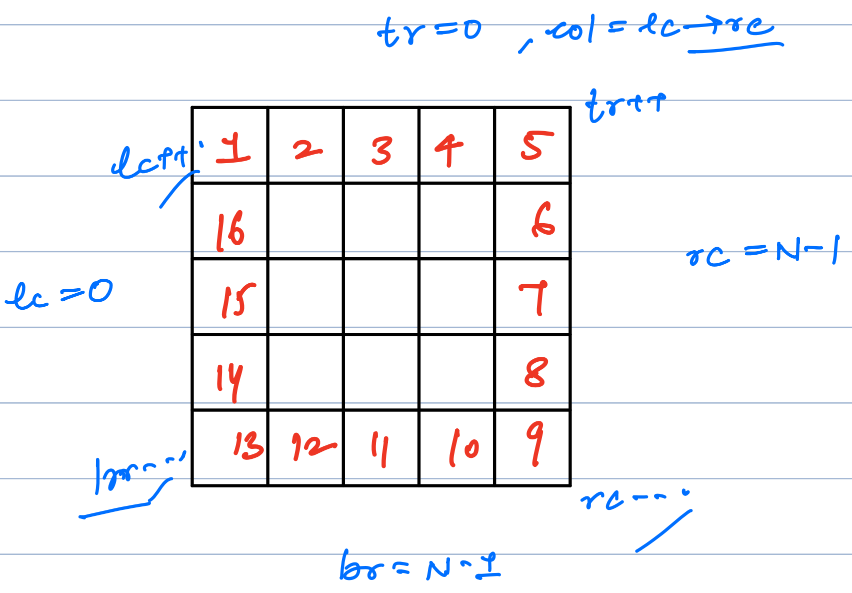
Spiral order Matrix

matrix [N][N] → 1 → N^2 spiral order





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



1	2	3	4	5
16				6
15				7
14				8
13	12	11	10	9

```
int   tr=0,  br = N-1,  lc=0,  rc=N-1;
              int x = 1;
        while( x<=N*N )
        {
              // top-row
              for( j = lc; j<=rc; j++)
              {       arr[ tr ][j] = x;
                             x++;
              }
              tr++;

              // right col
              for( i= tr; i<=br; i++)
              {       arr[i][rc] = x;
                             x++;
              }
              rc--;
              // bottom row, left col  —— finish it up
        }
```