Q.1 Given an array, find the max sum subarray of length K.

A =	3	9	4	-2	5	13	-7	8
	0	1	2	3	4	5	6	7

K=4

s	e	ans
0	3	14
1	4	16
2	5	20
3	6	9
4	7	19

i) brute force : go on all subarrays of length K, find sum of this subarray, overall best sum is final ans.

```
int solve (int [] A, int k) {

    int s=0, e=k-1;
    int ans= Integer.MIN_VALUE;

    while ( e < A.length )  {
        // sum of subarray from s to e
        int sum = 0;
        for (int i= s; i<= e; i++) {
            sum += A[i];
        }
        if (sum > ans) {
            ans = sum;
        }
        s++;
        e++;
    }

    return ans;
}
```

k = 3

$$A = \begin{array}{cccccc} [2 & 9 & 4 & -1 & 3 & 8] \\ 0 & 1 & 2 & 3 & 4 & 5 \end{array}$$

s	e	sum
0	2	0 to 2
1	3	1 to 3
2	4	2 to 4
3	5	3 to 5
4	6	

TC: $O(n^2)$

SC: $O(1)$

TC: no. of subarrays of length k in A[] * k

In n length array, how many k length subarrays can exist.

$$A = \begin{array}{cccccccc} [& 2 & 9 & 5 & 7 & 8 & 1 & 6 \,] \\ & 0 & 1 & 2 & 3 & 4 & 5 & 6 \end{array}$$

k	no. of subarrays of length k	
1	7	n
2	6	$n-1$
3	5	$n-2$

$$n - (k-1) = n-k+1$$

T C : no. of subarrays of length k in A[] * k

$$\begin{tikzcd}[column sep=small] & (n-k+1) * k \arrow[dl, "k=1"'] \arrow[d, "k=n"] \arrow[dr, "k=\frac{n}{2}"] & \\ (n-1+1) * 1 & (n-n+1) * n & (n-\frac{n}{2}+1) * \frac{n}{2} \\ O(n) & O(n) & O(n^2) \end{tikzcd}$$

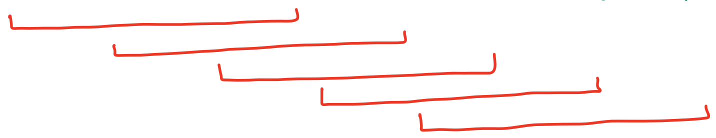
ii) Improvisation: prefix sum

```
int solve (int [] A, int k) {
    int s=0, e=k-1;
    int ans= Integer.MIN_VALUE;
    int [] ps = prefixSum(A);
    while ( e < A.length ) {
        // sum of subarray from s to e
        int sum = 0;
        if(s==0) {
            sum= ps[e];
        }
        else {
            sum= ps[e] - ps[s-1];
        }
        if(sum > ans) {
            ans= sum;
        }
        s++;
        e++;
    }
    return ans;
}
```

TC: O(N)

SC: O(N)

A =	3	9	4	-2	5	13	-7	8	k=4
	0	1	2	3	4	5	6	7	



carry forward on fixed length subarray → Sliding window

s	e	sum
0	3	$A[0] + A[1] + A[2] + A[3]$
1	4	$\cancel{A[0]} + A[1] + A[2] + A[3] - \cancel{A[0]} + A[4]$
2	5	$\cancel{A[1]} + A[2] + A[3] + A[4] - \cancel{A[1]} + A[5]$
3	6	$\cancel{A[2]} + A[3] + A[4] + A[5] - \cancel{A[2]} + A[6]$
4	7	$\cancel{A[3]} + A[4] + A[5] + A[6] - \cancel{A[3]} + A[7]$

$sum - A[s-1] + A[e]$

```
int  solve ( int [] A, int K ) {
    int sum = 0;
    // find the sum of first window
    for (int i=0; i<K; i++) {
            sum+=A[i];
    }

     int ans= sum;
     // apply sliding window
      int s =1 ,  e =K;
      while (e<n) {
            sum = sum - A[s-1] + A[e];
            if (sum > ans) {
                  ans= sum;
            }
            s++;
            e++;
      }
      return ans;
}
```

K = 3

$$\begin{array}{ccccccc} A = & [2 & 3 & -1 & 4 & 5 & 1] \\ & 0 & 1 & 2 & 3 & 4 & 5 \end{array}$$

ans= ~~4~~ ~~6~~ ~~8~~ 10

s	e	sum
		4
1	3	4 - 2 + 4 = 6
2	4	6 - 3 + 5 = 8
3	5	8 - (-1) + 1 = 10
4	6	

TC: O(n)

SC: O(1)

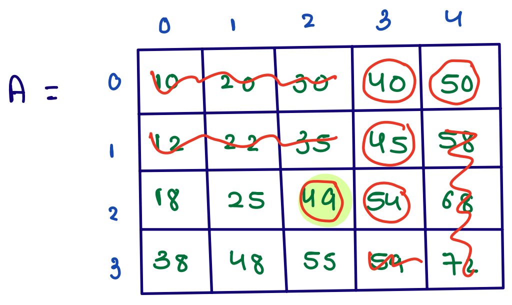
Q.2 Given a row and col wise sorted matrix, find if K is present in it or not.

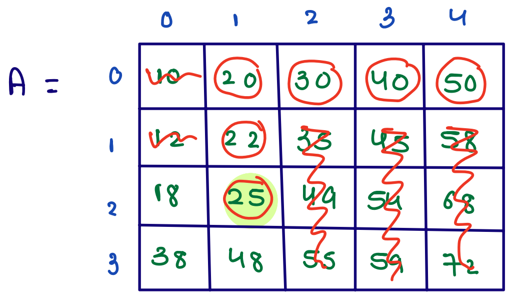
$$A = \begin{array}{c|ccccc} & 0 & 1 & 2 & 3 & 4 \\ \hline 0 & 10 & 20 & 30 & 40 & 50 \\ 1 & 12 & 22 & 35 & 45 & 58 \\ 2 & 18 & 25 & 49 & 54 & 68 \\ 3 & 38 & 48 & 55 & 59 & 72 \end{array}$$

$k = 49$

i) Brute force, searching the entire matric tc : $O(n*m)$

$K = 49$





$K = 25$

$K = 49$

$$A = \begin{array}{c|ccccc} & & & j & & \\ & 0 & 1 & 2 & 3 & 4 \\ \hline 0 & \cancel{10} & \cancel{20} & \cancel{30} & 40 & 50 \\ 1 & \cancel{12} & \cancel{22} & \cancel{35} & 45 & \cancel{58} \\ i \quad 2 & 18 & 25 & 49 & 54 & \cancel{68} \\ 3 & 38 & 48 & 55 & \cancel{59} & \cancel{72} \end{array}$$

```
boolean  search ( int [] [] A,  int  k ) {
    int  n= A.length;
    int  m= A[0].length;
    int  i= 0,  j = m-1;
    while ( i<n  &&  j>=0 )    {
        if (A[i][j] == K )  {
              return true;
        }
        else if (A[i][j] > K ) {
              j--;
        }
        else if (A[i][j] < K )  {
              i++;
        }
    }
    return false;
}
```

TC: O(N)

dry run

$K = 49$

```
while ( i<n && j>=0 )   {
    if (A[i][j] == K )  {
            return true;
    }
    else if ( A[i][j] > K ) {
            j--;
    }
    else if (A[i][j] < K ) {
            i++;
    }
}
```

$$\begin{array}{c|ccccc} & \multicolumn{5}{c}{j} \\ & 0 & 1 & 2 & 3 & 4 \\ \hline 0 & 10 & 20 & 30 & 40 & 50 \\ 1 & 12 & 22 & 35 & 45 & 58 \\ i\quad 2 & 18 & 25 & 49 & 54 & 68 \\ 3 & 38 & 48 & 55 & 59 & 72 \end{array}$$

$K = 47$

```
while ( i<n && j>=0 )   {
    if (A[i][j] == K )  {
            return true;
    }
    else if ( A[i][j] > K ) {
            j--;
    }
    else if (A[i][j] < K ) {
            i++;
    }
}
return false;
```

$$\begin{array}{c|ccccc} & \multicolumn{5}{c}{j} \\ & 0 & 1 & 2 & 3 & 4 \\ \hline 0 & \cancel{10} & \cancel{20} & \cancel{30} & 40 & 50 \\ 1 & \cancel{12} & \cancel{22} & \cancel{35} & 45 & \cancel{58} \\ 2 & \cancel{18} & 25 & 49 & 54 & \cancel{68} \\ 3 & 38 & 48 & \cancel{55} & \cancel{59} & \cancel{72} \end{array}$$

i

$i = \overset{4}{\cancel{3}} \quad j = 0$

Q.3 Given a 2D matrix of N x N, print its outermost boundary in clockwise direction.

$$A = \begin{array}{c|ccccc} & 0 & 1 & 2 & 3 & 4 \\ \hline 0 & 10 & 20 & 25 & 15 & 12 \\ 1 & 19 & 18 & 13 & 28 & 101 \\ 2 & 15 & 5 & 6 & 7 & 34 \\ 3 & 9 & 94 & 38 & 10 & 28 \\ 4 & 6 & 7 & 8 & 12 & 55 \end{array}$$

10 20 25 15 12 101

34 28 55 12 8 7

6 9 15 19

n = 5

$$\begin{array}{c|ccccc} & 0 & 1 & 2 & 3 & 4 \\ \hline 0 & 10 & 20 & 25 & 15 & 12 \\ 1 & 19 & 18 & 13 & 28 & 101 \\ 2 & 15 & 5 & 6 & 7 & 34 \\ 3 & 9 & 94 & 38 & 10 & 28 \\ 4 & 6 & 7 & 8 & 12 & 55 \end{array}$$

print n-1 values L to R

print n-1 values T to B

print n-1 values R to L

print n-1 values B to T

$$\begin{array}{c|ccccc} & 0 & 1 & 2 & 3 & 4 \\ \hline 0 & 10 & 20 & 25 & 15 & 12 \\ 1 & 19 & 18 & 13 & 28 & 101 \\ 2 & 15 & 5 & 6 & 7 & 34 \\ 3 & 9 & 94 & 38 & 10 & 28 \\ 4 & 6 & 7 & 8 & 12 & 55 \end{array}$$

$n=5$

```
void  boundary ( int [] [] A ) {
    int n= A.length;
    int i=0, j=0;
    // print n-1 values left to right
    for (int k=1; k<=n-1; k++) {
            sop(A[i][j] + " ");
            j++;
    }
    // print n-1 values top to bottom
    for (int k=1; k<=n-1; k++) {
            sop(A[i][j] + " ");
            i++;
    }
    // print n-1 values right to left
    for (int k=1; k<=n-1; k++) {
            sop(A[i][j] + " ");
            j--;
    }
    // print n-1 values bottom to top
    for (int k=1; k<=n-1; k++) {
            sop(A[i][j] + " ");
            i--;
    }
}
```

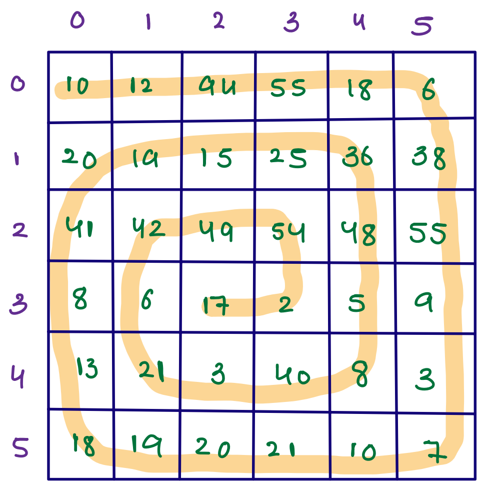
i	j	
0	0	
0	1	
0	2	
0	3	
0	4	out

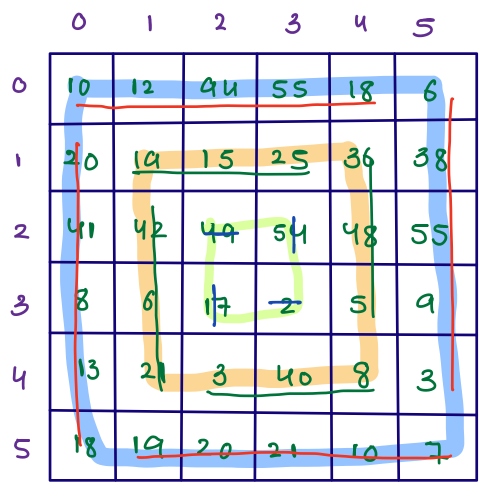
i	j	
0	4	
1	4	
2	4	
3	4	
4	4	out

i	j	
4	4	
4	3	
4	2	
4	1	
4	0	out

i	j	
4	0	
3	0	
2	0	
1	0	
0	0	out

Q.4 Given a 2D matrix of N x N, print it in spiral manner.





i	j	n
0	0	6
1	1	4
2	2	2
3	3	0

```
void  spiral ( int [ ] [ ] A ) {
        int n= A.length;
        int i=0, j=0;
        while ( n > 1)    {
            // print n-1 values left to right
            for (int k= 1; k <= n-1; k++) {
                    sop(A[i][j] + " ");
                    j++;
            }
            // print n-1 values top to bottom
            for (int k= 1; k <= n-1; k++) {
                    sop(A[i][j] + " ");
                    i++;
            }
            // print n-1 values right to left
            for (int k= 1; k <= n-1; k++) {
                    sop(A[i][j] + " ");
                    j--;
            }
            // print n-1 values bottom to top
            for (int k= 1; k <= n-1; k++) {
                    sop(A[i][j] + " ");
                    i--;
            }
            i++; j++;
            n= n-2;
        }
        if (n==1) {
                sop (A[i][j] + " ");
        }
}
```

	0	1	2	3	4
0	~~10~~	~~20~~	~~25~~	~~15~~	~~12~~
1	~~19~~	~~18~~	~~13~~	~~28~~	~~101~~
2	~~15~~	~~5~~	6	~~7~~	~~34~~
3	~~4~~	~~94~~	~~38~~	~~10~~	~~28~~
4	~~6~~	~~7~~	~~8~~	~~12~~	~~55~~

i	j	n
0	0	5
1	1	3
2	2	1

```
int  solve ( int [] A, int K) {
    int sum = 0;
    // find the sum of first window
    for (int i=0; i<K; i++) {
            sum += A[i];
    }

     int ans = sum;

     // apply sliding window

     for ( int s=1, e=K ; e < n ; s++, e++) {
           sum = sum - A[s-1] + A[e];

           if (sum > ans) {
                ans = sum;
           }

     }

     return ans;

}
```

writing using for loop.

K = 3

$$\begin{array}{ccccccc} A = [& 2 & 3 & -1 & 4 & 5 & 1] \\ & 0 & 1 & 2 & 3 & 4 & 5 \end{array}$$