* Yash is busy with some personal work & wouldn't be available for a couple of lectures.
* Will start by 9.06pm.

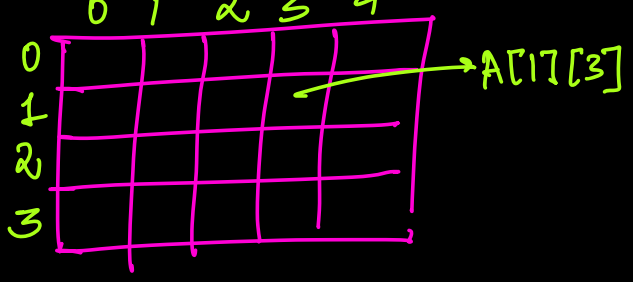
A(4x5)

0 3 0 4

```
int [][] A= new int[4][5];
```

rows cols.

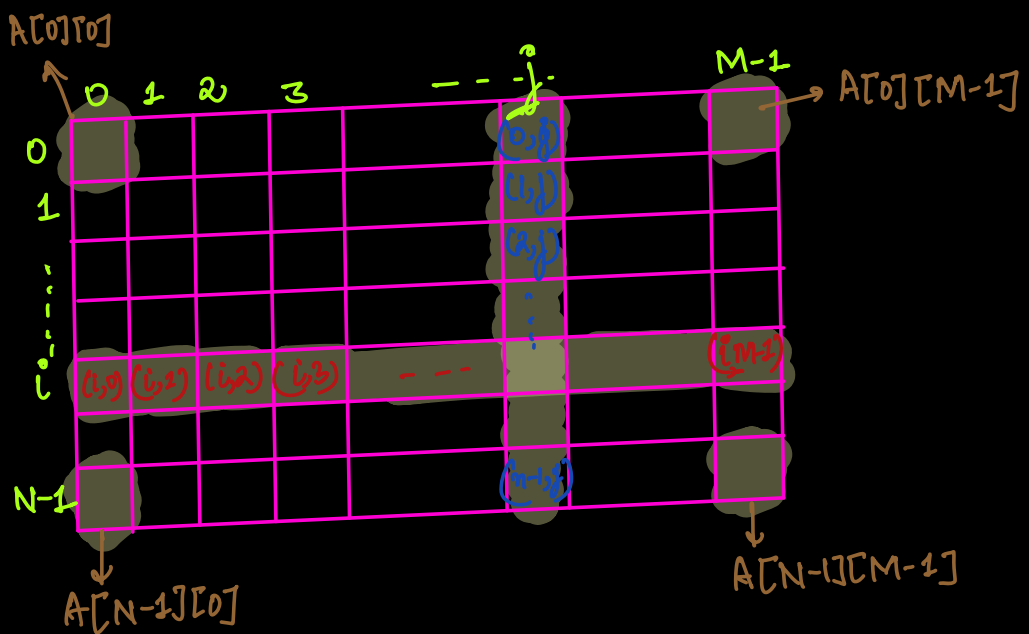
A



0 M-1

```
mat[N][M]
```

0 N-1



Obs :-

When iterating over a row :-
→ row index ✗
→ col index → [0, M-1]

When iterating over a col
→ col index ✗
→ row index → [0, N-1]

Q Given a matrix (N, M). Print rowwise sum.

I/P :- mat[3][4]

	0	1	2	3
0	4	3	1	7
1	6	2	3	4
2	5	3	2	7

O/P :-

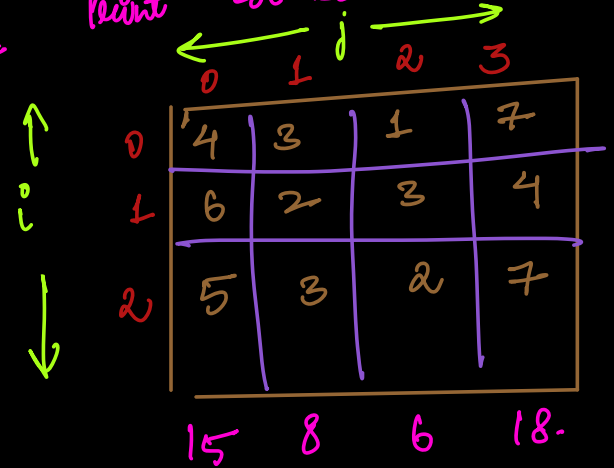
15
15
17.

```
void printRowWise( int [][] mat, int rows, int cols) {

    for( int i=0; i<rows; i++) {
        //now I need to calculate sum for ith row.
        int sum=0;
        for( int j=0; j<cols; j++) {
            sum = sum + mat[i][j];
        }
        print(sum);
    }

}
```

$O(M*N)$ ↓ rows

SC :- $O(1)$

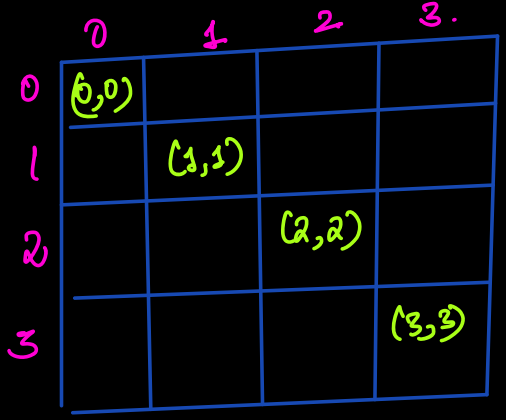
Q Print colwise sum.




```
for( int j=0; j<cols; j++){
    //need to calculate sum of jth col.
    int sum=0;
    for(int i=0; i<rows; i++){
        sum = sum + mat[i][j];
    }
    print(sum);
}
```

Q. Given a square matrix.
no. of rows = no. of cols.
Print all the diagonals.

mat[4][4].




```
for(int i=0; i<n; i++){
    for(int j=0; j<n; j++){
        if( i==j){
            print(mat[i][j]);
        }
    }
}
```

O(n²)


```
int i=0, j=0;
while( i<N && j<N ){
    print(mat[i][j]);
    i++; j++;
}
```

O(N) ←
S.C → O(1).

i,j → 0
mat[0][0].

i,j → 1
mat[1][1]

i,j → 2
mat[2][2],

i,j → 3
mat[3][3]

i,j → 4

Obs.

1. rowIndex ↑ → (by 1)
   colIndex → (by 1)

n x n.

i → 0
j → n-1

```
int i=0; int j= n-1;
while( i<n && j>=0 )
{
    print(mat[i][j]);
    i++;
    j--;
}
```

'i' is within bounds
'j' is within bounds

T.C → O(N)
S.C → O(1)

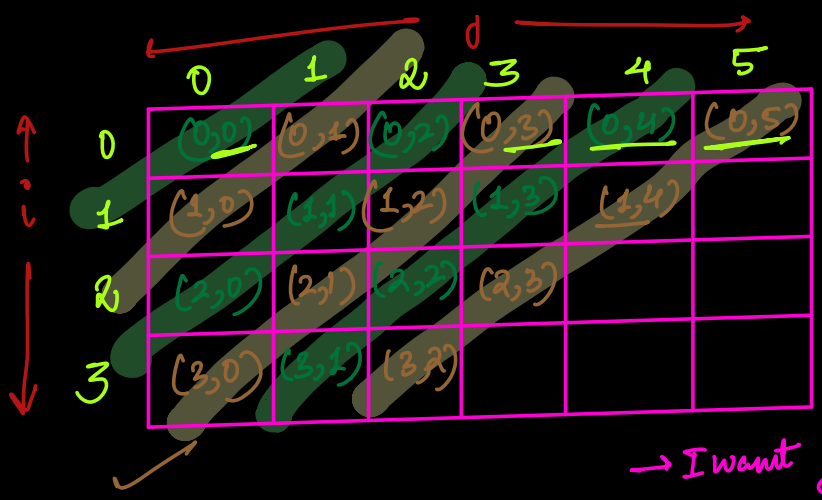
Q. Given a rectangular matrix. Iterate over diagonals. (Right → Left)

mat[N][M]

mat[4][6].

m → 6

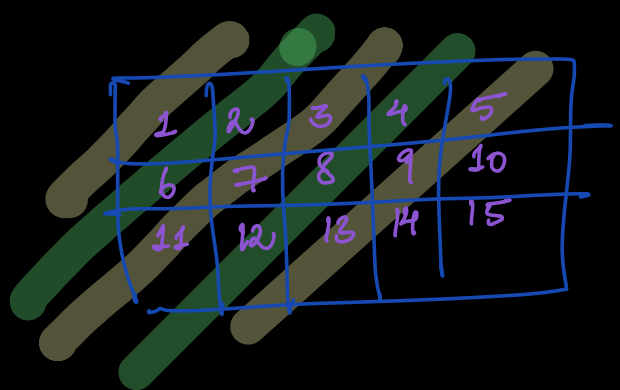
5 → m-1



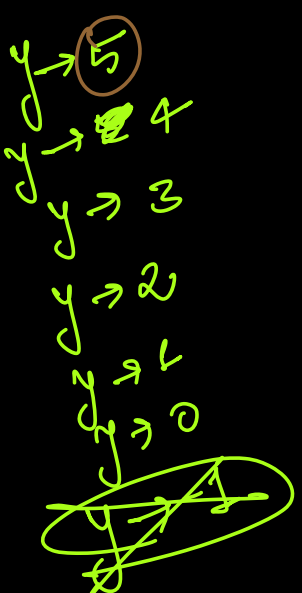
1st diagonal (i,j)
(0,5)
(1,4)
(2,3)
(3,2)
(4,0)

2nd diagonal (i,j)
(0,4)
(1,3)
(2,2)
(3,1)
(4,0)

→ I want i < n && j >= 0



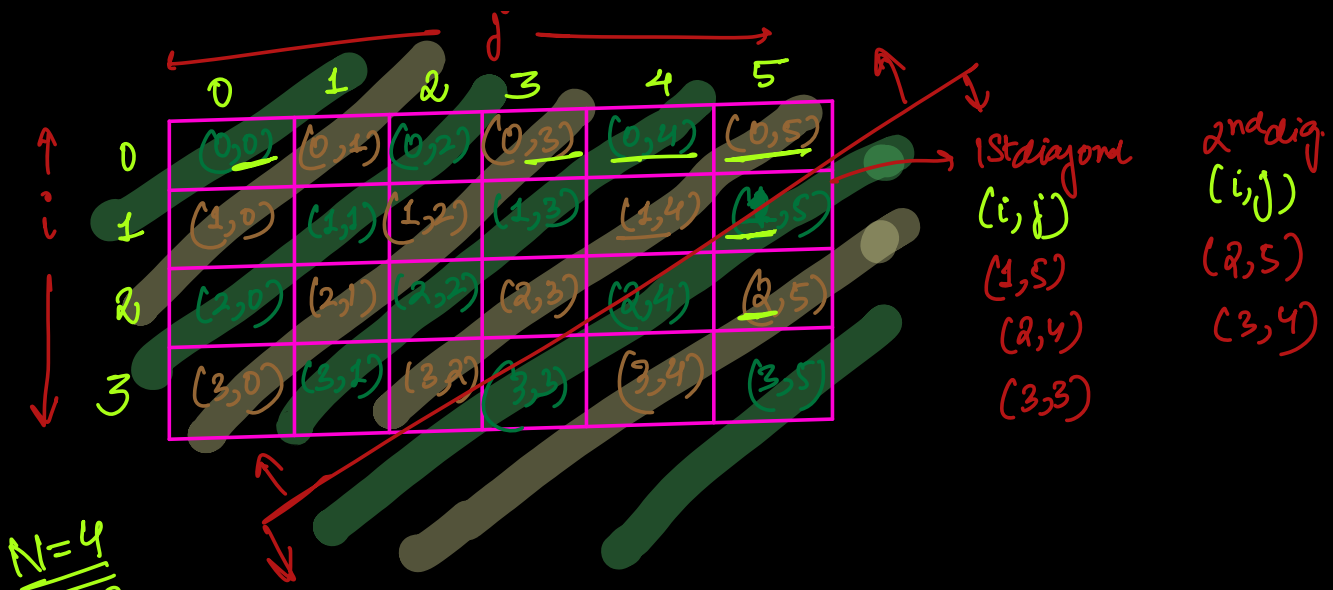
5 9 13
4 8 12
3 7 11
2 6
1




```
for(int y=m-1; y>=0; y--){
    int i=0
    int j=y
    // print diagonal with starting indices as (i,j)
    while( i<n && j>=0){
        print(mat[i][j]);
        i++;
        j--;
    }
}
```

1st diagonal (i,j)
(0,5) ← y
(1,4)
(2,3)
(3,2)
(4,0)

2nd diagonal (i,j)
(0,4) ← y
(1,3)
(2,2)
(3,1)
(4,0)



N=4

3?

x → 1
x → 2
x → 3

```
for(int x=1; x<N; x++) {

    int i = x
    int j = M-1   print diagonal with starting indices as (i,j)

    while( i<n && j>=0){
        print( mat[i][j]);
        i++;
        j--;
    }
}
```

1st diagonal (i,j)	2nd diagonal (i,j)
(0,5) ← y	(0,4) ← y
(1,4)	(1,3)
(2,3)	(2,2)
(3,2)	(3,1)
(4,1)	(4,0)

OVERALL CODE:

```
for(int y=m-1; y>=0; y--){
    int i=0
    int j=y   ] print diagonal with starting indices as (i,j)
    while( i<n && j>=0){
        print(mat[i][j];
        i++;
        j--;
    }
}
```

T.C → O(M+N)
↓
col rows.

S.C → O(1)

```
for(int x=1; x<N; x++){
    int i=x
    int j=M-1  print diagonal with starting indices as (i,j)
    while( i<n && j>=0){
        print(mat[i][j];
        i++;
        j--;
    }
}
```

Returns

Q. Given a Mat[N][N]. ~~Calculate~~ Return Transpose

→ S.C → O(1)

	0	1	2	3	4
0	1	2	3	4	5
1	6	7	8	9	10
2	11	12	13	14	15
3	16	17	18	19	20
4	21	22	23	24	25

	0	1	2	3	4
0	1	6	11	16	21
1	2	7	12	17	22
2	3	8	13	18	23
3	4	9	14	19	24
4	5	10	15	20	25

$x(i,j) \rightleftharpoons x(j,i)$

$\rightleftharpoons (3,3)$

$6(1,0) \rightleftharpoons 6(0,1)$

$18(3,2) \rightleftharpoons 18(2,3)$

i,j

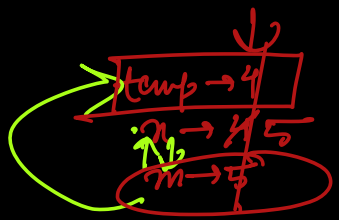
```
for(int i=0; i<n; i++){          → upper Δ
    for(int j=i+1; j<n; j++){
        swap(arr[i][j], arr[j][i]);
    }
}
```

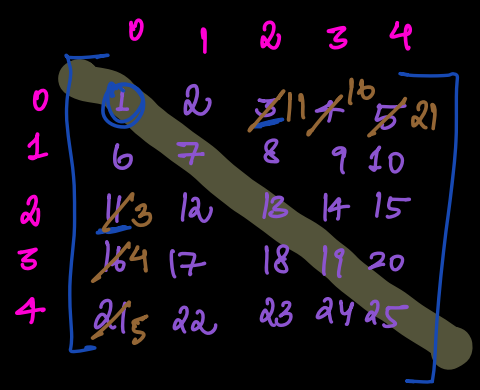
iterate in upper Δ or lower Δ.

S.C = O(1)

```
swap(n, m){
    temp = n;
    n = m;
    m = temp;
}
```

4 5





i → 0 to n-1
j → 0 to n-1

i=0, j → [0,1,2,3,4]

a[0][1] → a[1,0]
a[0][2] → a[2][0]

i=1, j → [0,1,2,3,4]

a[1][0] → a[0][1]
2 → 6

Rectangular matrix. (you will need space)

A (2x5)

	0	1	2	3	4
0	1	2	3	4	5
1	6	7	8	9	10

A(i,j) → AT(j,i)

AT (5x2)

	0	1
0	1	6
1	2	7
2	3	8
3	4	9
4	5	10

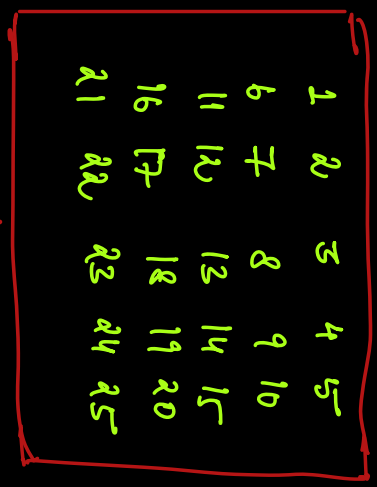
A(nxm)

```
transpose = new int[m][n];
for(int i=0; i<n; i++){
    for(int j=0; j<m; j++){
        t[j][i] = arr[i][j];
    }
}
```

→ S.C

Q. Given a mat NXN.
Rotate it by 90° 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



* Think in terms of transpose.
(of given array).