Arrays: 2D Matrices

DEFINATION

↳ 2D matrices is specific type of 2D array that has rectangular grid of numbers, where each number is called an element.

⇨ Declaration

int mat[N][M];

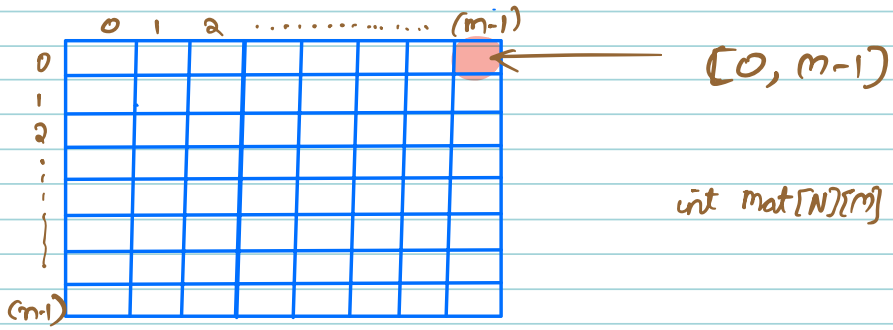
- int → Data type
- mat → Variable Name
- N → rows
- M → columns

	0	1	2
0	(0,0)	(0,1)	(0,2)
1	(1,0)	(1,1)	(1,2)
2	(2,0)	(2,1)	(2,2)

3 × 3

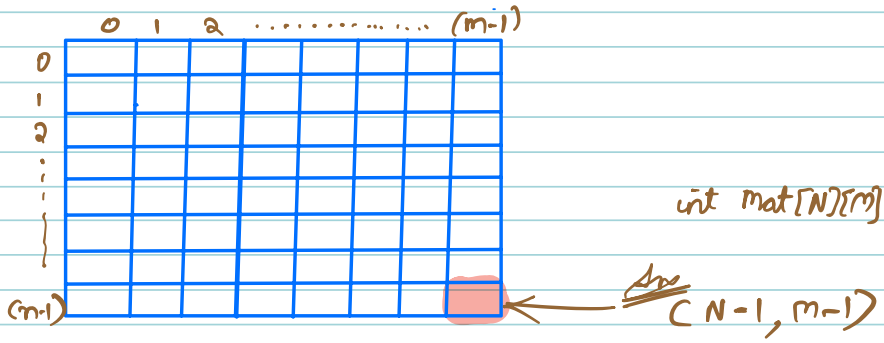
mat[N][M] → N×M elements

Quiz 1: Given a matrix of size N×M. what will be the index of the top right cell?



Ans:- (0, m-1)

Quiz 2: Given a matrix of size N × M, what will be the index of the bottom right cell?



Question :- Given 2D matrix mat[N][M], print row-wise sum.

O/P

Mat[3][4] =

1	2	3	4	→ 10
5	6	7	8	→ 26
9	10	11	12	→ 42

PSEUDO CODE

```
for (i = 0; i < row; i++){
    Sum = 0;
    for (j = 0; j < column; j++){
        Sum += mat[i][j];
    }
    print(Sum);
}
```

SOP

Quiz :- what is TC & SC of finding row wise Sum for matrix [N][M].

TC → O(N*M)

SC → O(1)

Question :- Given 2D matrix mat[N][M], print column-wise sum.

Mat[3][4] =

1	2	3	4
5	6	7	8
9	10	11	12

O/P 15 18 21 24

PSEUDO CODE

```
for ( col=0 ; col < columns ; col++){
    Sum = 0;
    for ( row = 0 ; row < rows ; row++){
        Sum += Mat[row][col];
    }
    Print (Sum);
}
```

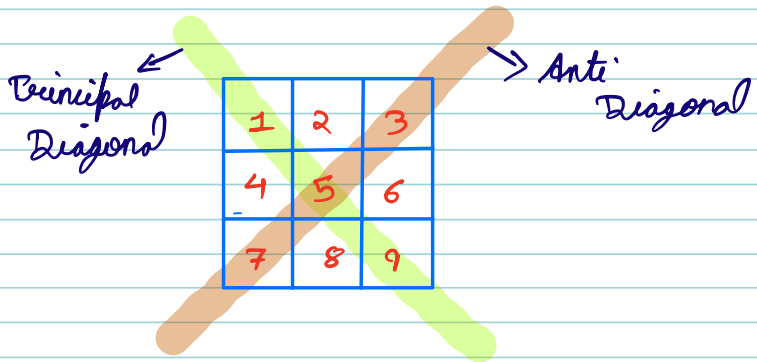
TC → O(N x M)

SC → O(1)

Question 3:- Given a 2D square matrix mat[N][N] Print :-

1:) Principal Diagonal :- From top left to bottom right

2:) Anti Diagonal :- From top right to bottom left



(1) Print Principal Diagonal

Observation

↳ i & j is same & always increasing

PSEUDO CODE

```
i = 0
while (i < N) {
    Print (mat[i][i]);
    i++;
}
```

Quiz 4 : what is the time complexity of printing Diagonal elements ?

TC → O(N)

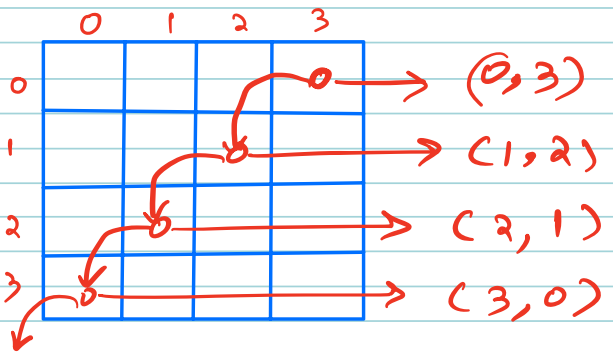
SC → O(1)

② Print Anti Diagonal

observation

↳ Let say the element is represent like (i, j) diagonal, Then we can say for anti we can say i is increasing & j is decreasing

Ex :-



PSEUDO CODE

only i is enough

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

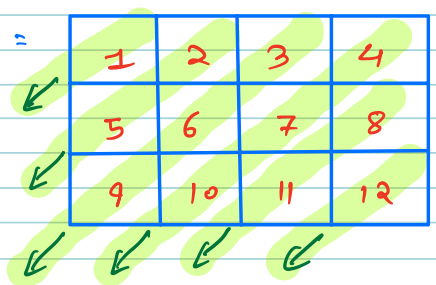
TC → O(N)

SC → O(1)

Question :- Given a 2D matrix, print all the elements diagonally from right to left.

Ex :-

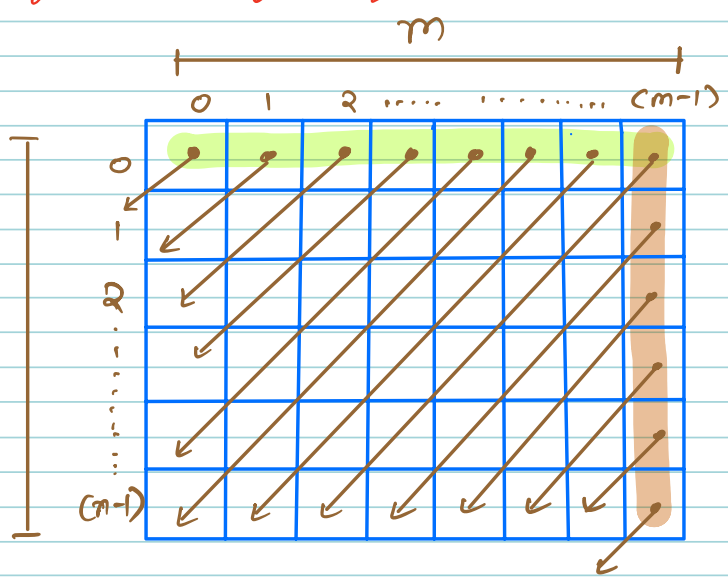
mat[3][4] =



Output :-

1
2 5
3 6 9
4 7 10
8 11
12

Quiz 5 : Given a matrix of size N * M, how many Right to Left diagonals will be there ?



Total Diagonals = N + M - 1

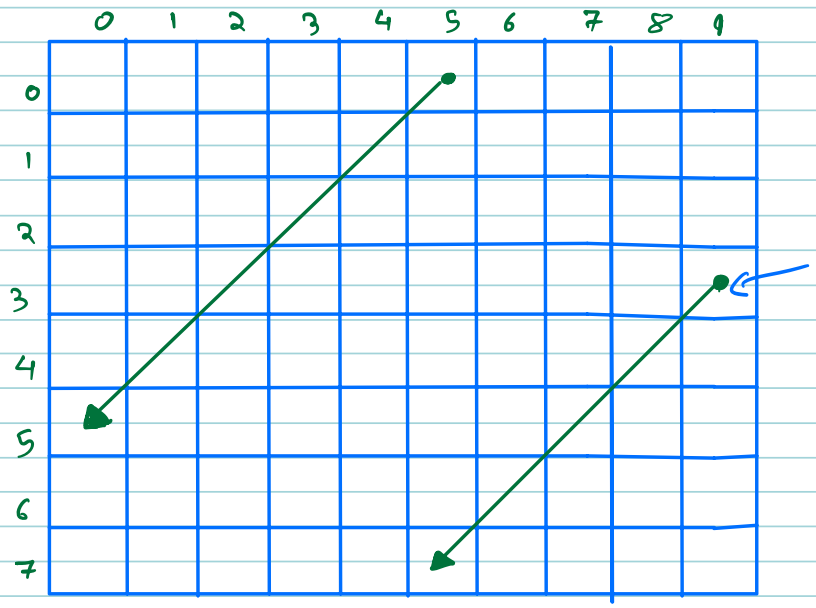
we are subtracting because (0, m-1) is repeated

Ques 6 :- Given a matrix of size 4 x 5, how many Right to Left diagonal will be there?

int Mat [4][5]

Total Diagonals = 4 + 5 - 1
= 8

=> Back to question



Ex (0,5) → (1,4) → (2,3) → (3,2)
↓
(6,-1) ← (5,0) ← (4,1)

Ex (3,9) → (4,8) → (5,7) → (6,6)
↓
(8,4) ← (7,5)

PSEUDO CODE

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

↳ The above code will print diagonal given starting point

observations for Starting point

① For First 'n' diagonal the starting point will be in the first row.

② For last 'n-1' diagonal the starting point will be last Column.

PSEUDO CODE

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

Obs! →

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

Quiz 7 : Time complexity of above solution ??

TC → (N * M)

O(N * M)

SC → O(1)

Question :- Given a square 2D Matrix [N][N], find transpose.

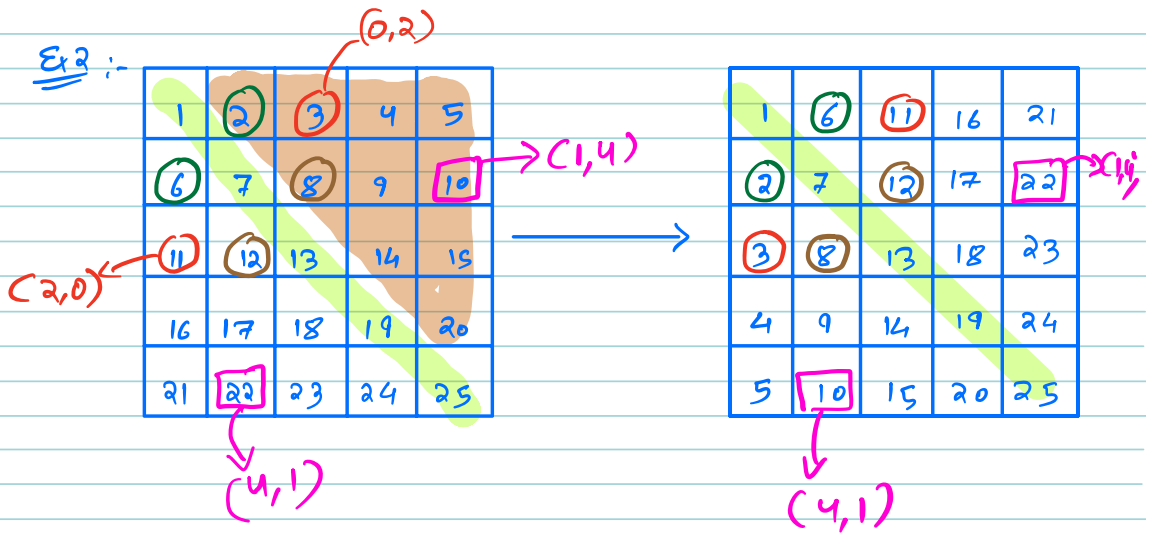
Ex:-

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

Row becomes column, col becomes Row →

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

Ex2 :-



Observations

1. Principal diagonal is not changing.
2. Mat [i] [j] ⟷ Mat [j] [i]

PSEUDO CODE

```
for (i = 0; i < N; i++){
    for (j = 0; j < M; j++){
        swap ( Mat[i][j], Mat[j][i])
    }
}
```

★ The above code is ineffective because it will swap twice.

★ We can correct the code by just iterating through upper half triangle

PSEUDO CODE

```
for (i = 0; i < N; i++){
    for (j = i+1; j < M; j++){
        swap ( mat[i][j], mat[j][i])
    }
}
```

Quiz :- Time & Space complexity of above code

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

i	j	Total iterations
0	[1 N-1]	N-1
1	[2 N-1]	N-2
2	[3 N-1]	N-3
		⋮
N-1	[N N-1]	0

Total iterations = $0 + 1 + 2 + \dots + (N-2) + (N-1)$

By sum of n natural nos

$$\frac{n * (n+1)}{2}$$

when $n = N-1$

$$\frac{(N-1) * N}{2}$$

TC → $O(N^2)$

SC → $O(1)$

Question :- Given a matrix [N][N], rotate it to 90° clockwise

Ex:-

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

After Rotate →

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

Ex

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

After Rotating 90° →

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

↓ transpose.

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

↗ Reverse

So we can say

→ Transpose + Reversal = rotate 90°

TC $\rightarrow 2N^2 \approx O(N^2)$

SC $\rightarrow O(1)$