Today's Quote ⇒

EVERYTHING IS HARD BEFORE IT IS EASY.

Today's content

- → Basics
- → Problems

] 2-D array or matrices.

How to declare?

rows : horizontal lines

int mat [4] [5]

columns : vertical lines

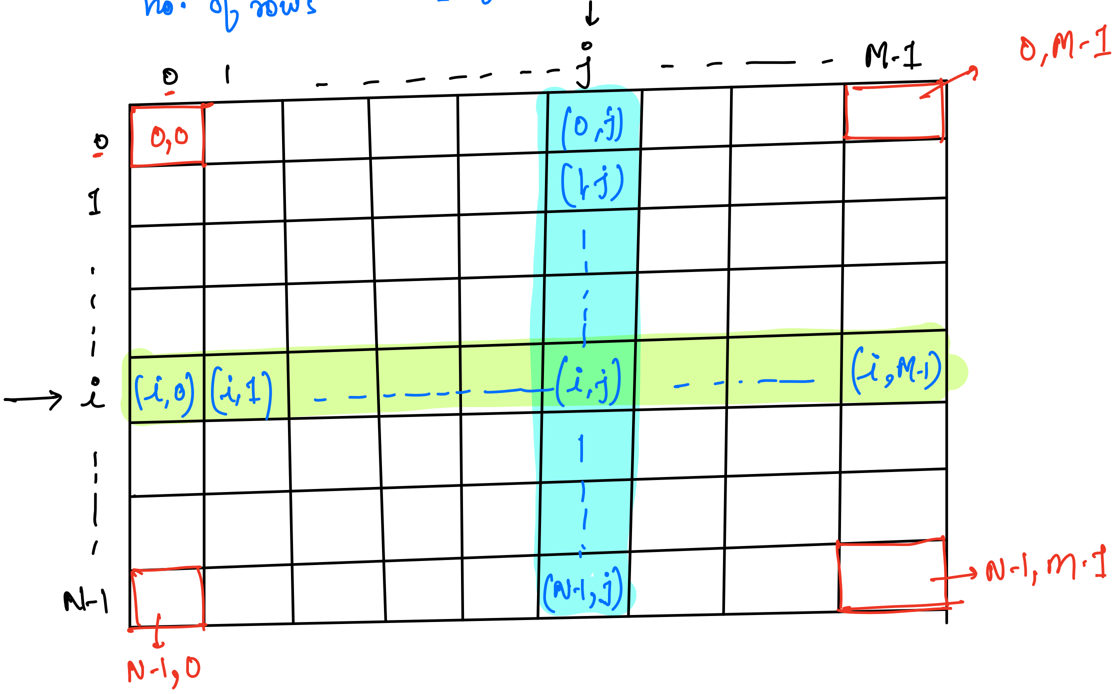
|   | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 |   |   |   |   |   |
| 1 |   |   |   |   |   |
| 2 |   |   |   |   |   |
| 3 |   |   |   |   |   |

total no. of elements = rows * cols
= 4x5
= 20.

int mat [N] [M]

no. of rows

no. of columns.



observation 1 : If we move in $i^{th}$ - row

col changes [0 → M-1]

observation 2 : If we move in $j^{th}$ - col

row changes [0 → N-1]

Q.) Given mat[N][M], print row-wise sum.

Eg→ mat[3][4]

| | 0 | 1 | 2 | 3 | | sum. |
|---|---|---|---|---|---|---|
| 0 | 4 | 3 | 1 | 7 | : | 15 |
| 1 | 6 | 2 | 3 | 4 | : | 15 |
| 2 | 5 | 3 | 2 | 7 | : | 17 |

```
void printSum( arr, N, M) {
    for( i = 0 ; i < N ; i++) {
        sum = 0
        for( j = 0 ; j < M ; j++) {
            sum += arr[i][j]
        }
        // print sum.
    }
}
```

T.C → O(N*M)
S.C → O(1)

Given mat[N][M], print col wise sum ⇒ {# todo}

Eg→ 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 8 6 18

Q) Given square mat[N][N]. print diagonals → left to right
→ right to left

Eg: mat[4][4]

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | (0,0) | | | |
| 1 | | (1,1) | | |
| 2 | | | (2,2) | |
| 3 | | | | (3,3) |

```
i = 0, j = 0
while (i < N && j < N ){
        print( arr[i][j])
          i += 1
          j += 1
}
```

T.C → O(N) , S.C → O(1)

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | | | | 0,3 |
| 1 | | | 1,2 | |
| 2 | | 2,1 | | |
| 3 | 3,0 | | | |

0,3
↓
1,2
↓
2,1
↓
3,0
↓
4,-1

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

T.C → O(N) , S.C → O(1)

→ All squares are rectangle. ✓

→ All rectangles are square. ✗

Q: Given a mat [N][M], print all diagonals going from R → L.

diagonals starting from 0th row or M-1th column.

mat [4][6]

|   | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 |   |   | 0,2 |   | 0,4 |   |
| 1 |   | 1,1 |   | 1,3 |   | 1,5 |
| 2 | 2,0 |   | 2,2 |   | 2,4 |   |
| 3 |   | 3,1 |   | 3,3 |   |   |

| i, j | i, j |
|---|---|
| (0, 4) | (1, 5) |
| (1, 3) | (2, 4) |
| (2, 2) | (3, 3) |
| (3, 1) | (4, 2) x<br>stop. |
| (4, 0)<br>stop. |   |

mat [3][5]

|   | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |
| 1 | 6 | 7 | 8 | 9 | 10 |
| 2 | 11 | 12 | 13 | 14 | 15 |

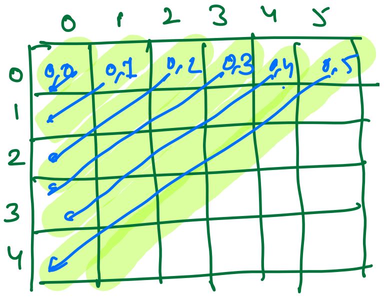
Output.

```
[ 1
  2  6
  3  7  11
  4  8  12
  5  9  13
  10 14
  15      ]
```

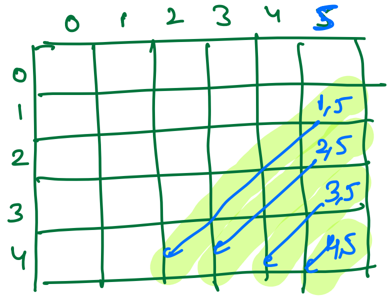
## Pseudo code

```
void printDiagonals( mat[][] , N , M){
    // print all diagonals starting from 0th row.
        for( j = 0 ; j < M ; j++){
            r = 0 , c = j
            while (r < N && c >= 0){
                Print(arr[r][c])
                r += 1 , c -= 1
            }
        }
    // print all Diagonals starting from M-1th column
        for( i = 1 ; i < N ; i++){
            r = i , c = M-1
            while (r < N && c >= 0){
                Print(arr[r][c])
                r += 1 , c -= 1
            }
        }
}
```

r = 0, c = ~~2~~ ~~3~~ ~~4~~ 5



r = ~~1~~ ~~3~~ 2 4 c = 5



$$\boxed{T.C \rightarrow O(N*M) , \quad S.C \rightarrow O(1)}$$

⇓

[We are touching all elements only once.]

Break 10 minutes

9:56 PM.

Q) Given matrix [N][N]. Calculate transpose of mat[] with S.C → O(1).

**Note → get transpose in the given matrix itself.**

**mat[5][5]:**

$$\begin{array}{c|ccccc} & 0 & 1 & 2 & 3 & 4 \\ \hline 0 & 1 & 2 & 3 & 4 & 5 \\ 1 & 6 & 7 & 8 & 9 & 10 \\ 2 & 11 & 12 & 13 & 14 & 15 \\ 3 & 16 & 17 & 18 & 19 & 20 \\ 4 & 21 & 22 & 23 & 24 & 25 \end{array}$$

rol 0 ← col 0
row 1 ← col 1
row 2 ← col 2
row 3 ← col 3
row 4 ← col 4

$$\begin{array}{c|ccccc} & 0 & 1 & 2 & 3 & 4 \\ \hline 0 & 1 & 6 & 11 & 16 & 21 \\ 1 & 2 & 7 & 12 & 17 & 22 \\ 2 & 3 & 8 & 13 & 18 & 23 \\ 3 & 4 & 9 & 14 & 19 & 24 \\ 4 & 5 & 10 & 15 & 20 & 25 \end{array}$$

idea : swap upper half elements with lower half elements.

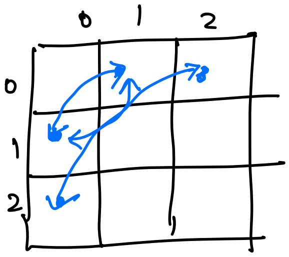
```
void   takeTranspose ( arr, N) {
    for ( i = 0 ; i < N ; i++) {
        for( j = i+1 ; j < N ; j++){
            //swap arr(i)[j] with arr[j][i]
            temp = arr(i][j]
            arr[i][j] = arr[j][i]
            arr [j] [i] = temp
        }
    }
}
```

|  | j |
|---|---|
| i = 0 | [1, N-1] |
| i = 1 | [2, N-1] |
| i = 2 | [3, N-1] |

**T.C → O(N²) , S.C → O(1)**

```
void    takeTranspose ( arr, N) {
    for ( i = 0 ; i < N ; i++) {
        for ( j = 0 ; j < N ; j++) {
            //swap arr[i][j] with arr[j][i]
            temp = arr[i][j]
            arr[i][j] = arr[j][i]
            arr[j][i] = temp
        }
    }
}
```

N=3.



arr[0][0] ← arr[0][0]

arr[0][1] ↔ arr[1][0]

arr[1][0] ← arr[0][1]

⇒ [Matrix is going to remain as it is]

// If rectangle We need to have extra space.

mat[2][5]

$$\begin{array}{c|ccccc} & 0 & 1 & 2 & 3 & 4 \\ \hline 0 & 1 & 2 & 3 & 4 & 5 \\ 1 & 6 & 7 & 8 & 9 & 10 \end{array}_{(2*5)} \xrightarrow{\text{transpose}} \begin{bmatrix} 1 & 6 \\ 2 & 7 \\ 3 & 8 \\ 4 & 9 \\ 5 & 10 \end{bmatrix}_{\underline{5\times 2}}$$

Q) Given a square matrix. Rotate 90' clockwise. [S.C → O(1)]

$$\begin{array}{c|ccccc} & 0 & 1 & 2 & 3 & 4 \\ \hline 0 & 1 & 2 & 3 & 4 & 5 \\ 1 & 6 & 7 & 8 & 9 & 10 \\ 2 & 11 & 12 & 13 & 14 & 15 \\ 3 & 16 & 17 & 18 & 19 & 20 \\ 4 & 21 & 22 & 23 & 24 & 25 \end{array}$$

0th row ↔ 4th col

1st row ↔ 3rd col

2nd row ↔ 2nd col

3rd row ↔ 1st col

4th row ↔ 0th col.

O/P.

$$\begin{array}{c|ccccc} & 0 & 1 & 2 & 3 & 4 \\ \hline 0 & 21 & 16 & 11 & 6 & 1 \\ 1 & 22 & 17 & 12 & 7 & 2 \\ 2 & 23 & 18 & 13 & 8 & 3 \\ 3 & 24 & 19 & 14 & 9 & 4 \\ 4 & 25 & 20 & 15 & 10 & 5 \end{array}$$

↓ transpose

Reverse every row.

$$\begin{array}{c|ccccc} & 0 & 1 & 2 & 3 & 4 \\ \hline 0 & 1 & 6 & 11 & 16 & 21 \\ 1 & 2 & 7 & 12 & 17 & 22 \\ 2 & 3 & 8 & 13 & 18 & 23 \\ 3 & 4 & 9 & 14 & 19 & 24 \\ 4 & 5 & 10 & 15 & 20 & 25 \end{array}$$

reverse 0th row

reverse 1st row

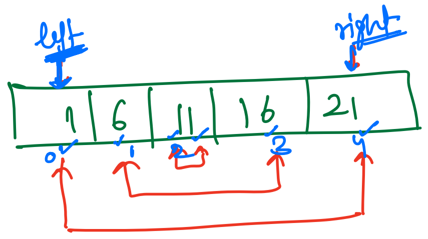
reverse 2nd row

reverse 3rd row

reverse 4th row

$$\begin{array}{c|ccccc} & 0 & 1 & 2 & 3 & 4 \\ \hline 0 & 21 & 16 & 11 & 6 & 1 \\ 1 & 22 & 17 & 12 & 7 & 2 \\ 2 & 23 & 18 & 13 & 8 & 3 \\ 3 & 24 & 19 & 14 & 9 & 4 \\ 4 & 25 & 20 & 15 & 10 & 5 \end{array}$$

1-D array.



// step-1. take transpose of the given matrix. → $N^2$

// Step-2. Reverse every row.

```
for ( i = 0 ; i < N ; i++) { //reverse ith row
        left = 0 , right = N-1
        while ( left < right ) {
            //swap arr[i][left] with arr[i][right]
            temp = arr[i][left]
            arr[i][left] = arr[i][right]
            arr[i][right] = temp.
            left += 1
            right -= 1
        }
}
```

→ $N^2$

$$\boxed{\begin{array}{l} T.C \to O(N^2) \\ S.C \to O(1) \end{array}}$$

✓ [ 90°, 180°, 270°, 360°, – – ].

## Rotate Rectangular matrix.

{We need to have extra space}

$$\begin{array}{c|ccccc} & 0 & 1 & 2 & 3 & 4 \\ \hline 0 & 1 & 2 & 3 & 4 & 5 \\ 1 & 6 & 7 & 8 & 9 & 10 \end{array}$$

(2X5)

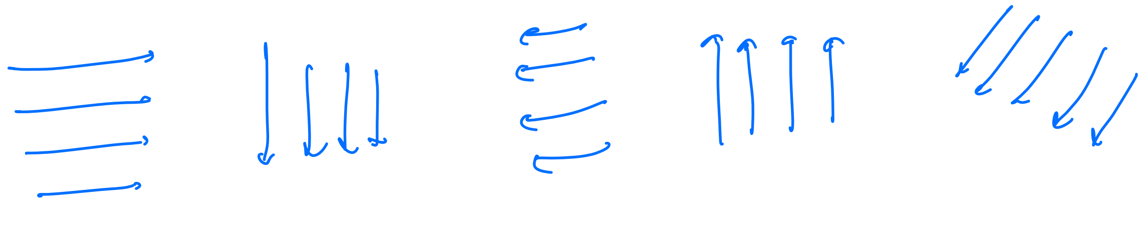
$$\begin{array}{c|cc} & 0 & 1 \\ \hline 0 & 6 & 1 \\ 1 & 7 & 2 \\ 2 & 8 & 3 \\ 3 & 9 & 4 \\ 4 & 10 & 5 \end{array}$$

(5X2)

{Todo}.

---

2-D → implementation- based.



no of rows. → arr.length , no. of columns → arr[0]. length.

$x = n(n+1)/2$

```
int[][] arr = new int[X][];
{
    arr[0] = new int[5];
    arr[2] = new int[3];
}
```

→ [Questions that are least solved by your batch.] 4.

→ [optional class] → attendance will not be counted.

→ Duration [2 -3 hours].

```
{ {5, 4, 3, 2, 1},
  {1, 11, 6},
  {10},
  {17, 18, 19, 20} }
```

N = 4.

```
int[][] arr = new int[4][ ];

arr[0] = new int[5];
arr[1] = new int[3];
arr[2] = new int[1];
arr[3] = new int[4];
```

Google.