Definition

A 2D matrix is a specific type of 2D array that has a rectangular grid of numbers where each number is called an element. It is a mathematical structure that consists of a set of numbers arranged in rows & columns.

Declaration

name of the matrix

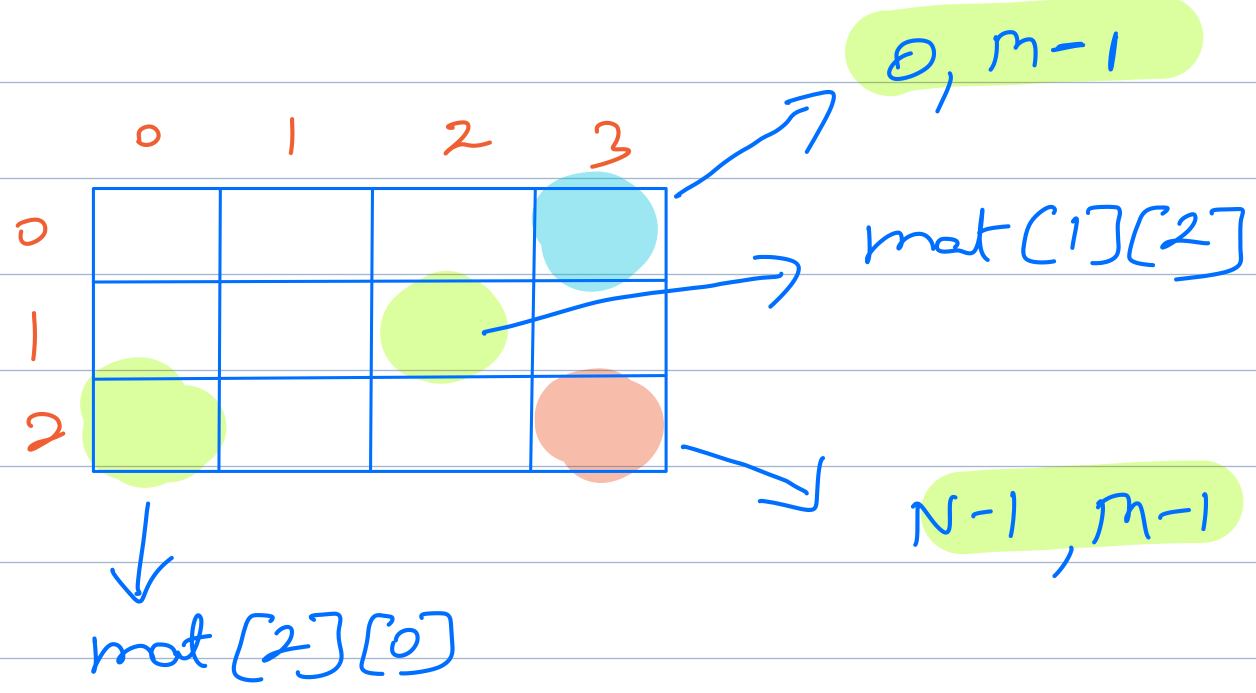
cols.

int mat[N][M];

datatype

rows

Ex:- int mat[3][4]



Quiz 1

0, M-1

Quiz 2

Q:- 1. Given a matrix, print row-wise sum.

ex:- mat[3][4]

	0	1	2	3
0	1	2	3	4
1	5	6	7	8
2	9	10	11	12

Output

10
26
42

Approach :

```
void    rowSum ( int  mat[][]
                          , N, m) {
        int sum;
    for (row = 0; row < N; row++){
                sum = 0;

        for(col = 0; col < m; col++) {
                sum += mat[row][col];
        }
        print (sum);
    }
}
```

T.C → O(N*m)

S.C → O(1)

Q:- Given a matrix, print col-wise sum.

ex:-

	0	1	2	3
0	1	2	3	4
1	5	6	7	8
2	9	10	11	12

Output
15
18
21
24

```
void      colSum ( int mat [] [], N, m) {
    for (col = 0 ; col < m ; col ++) {
            int   sum = 0;

        for (row = 0 ; row < N; row++){
            sum += mat[row][col];
        }
            print (sum);
    }
}
```

T.C → O(N*m)
S.C → O(1)

Q:- Given a square matrix, print its diagonals

m = N

0,0

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

N-1, N-1

Principal Diagonal

0, N-1

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

N-1, 0

Anti Diagonal

Approach

```
void  print Diagonals ( int mat[][] , N  ){

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

row	col
0	N-1
1	N-1 -1
2	N-1 -2
⋮	
N-1	N-1 - (N-1) 0

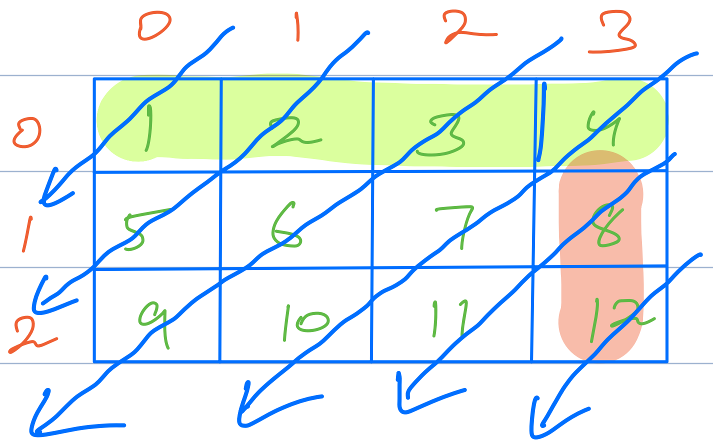
col = N-1 - row

```
      int  i = 0
      while ( i < N )      {
      |   print ( mat [ i ][ N-1-i ];
      |      i++;
      }
```

Quiz 4

T.C → O(N) ; S.C → O(1)

Q:- Print diagonals in a matrix (right to left)



$$\begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & 1 & 2 & 3 & 4 \\ 1 & 5 & 6 & 7 & 8 \\ 2 & 9 & 10 & 11 & 12 \end{array}$$

Output:

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

row++;
col--;

1,3
↓
2,2
↓
~~3~~, 1

Quiz 5

M + N - 1

Approach:

0 ~~0~~ ~~1~~ 2

i = ~~0~~; j = ~~2~~ ~~1~~ ~~0~~ -1

~~1~~ ~~2~~ 3

```
void printDiagonals (mat[][], N, m){
    int row = 0;
    for (col = 0; col < m; col++) {
        i = row; j = col; // Fixed the S.P.
        while (i < N && j >= 0 ) {
            print (mat[i][j]);
            i++;
            j--;
        }
        print ("\n");
    }
}
```

Output.

```
1
2 5
3 6 9
```

int col = m-1;

```
for (row = 1; row < N; row++) {
    i = row; j = col;  // Fixed the S.P.
    while (i < N && j >= 0) {
        print (mat[i][j]);
        i++;
        j--;
    }
    print("\n");
}
```

T.C → O(N * m)

S.C → O(1)

8:23

Q:- Transpose of a square matrix

rows → columns.

columns → rows.

$$\begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 1 & 2 & 3 \\ 1 & 4 & 5 & 6 \\ 2 & 7 & 8 & 9 \end{array} \quad \begin{array}{c} \text{row}(0) - \text{col}(0) \\ \longrightarrow \\ \text{row}(1) - \text{col}(1) \\ \longrightarrow \\ \text{row}(2) \\ \longrightarrow \\ \to \text{col}(2) \end{array} \quad \begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 1 & 4 & 7 \\ 1 & 2 & 5 & 8 \\ 2 & 3 & 6 & 9 \end{array}$$

mat (i](j] → mat(j](i];

Approach

```
for (row = 0; row < N ; row++) {
    for (col = row ; col < m; col++){
        swap (mat (row](col], mat(col](row]);
    }
}
```

N

N-1

N-2.

:

1

$3 * 4.$ → $4 * 3.$

Quiz 8

T.C → $O(N^2)$.

S.C → $O(1)$

Q:- Rotate a matrix 90° clockwise

	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

90° → clockwise rotation.

	0	1	2	3	4
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

↓ transpose.

	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

↑ reverse each row

vector<vector<int>> a

```
int mat[][] rotate90 ( mat[][] , N) {
    mat = transpose(mat[][], N);
    for ( row = 0; row < N; row++) {
        reverse(mat[row]);
    }
    return mat;
}
```

T.C → $O(N^2)$

S.C → $O(1)$.

Next Class :-

Memory Management

void reverse (int mat [