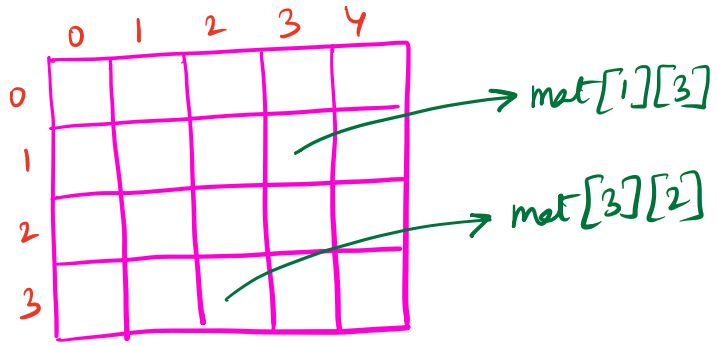
"There are no big problems, there are just a lot of little problems."

— Henry Ford

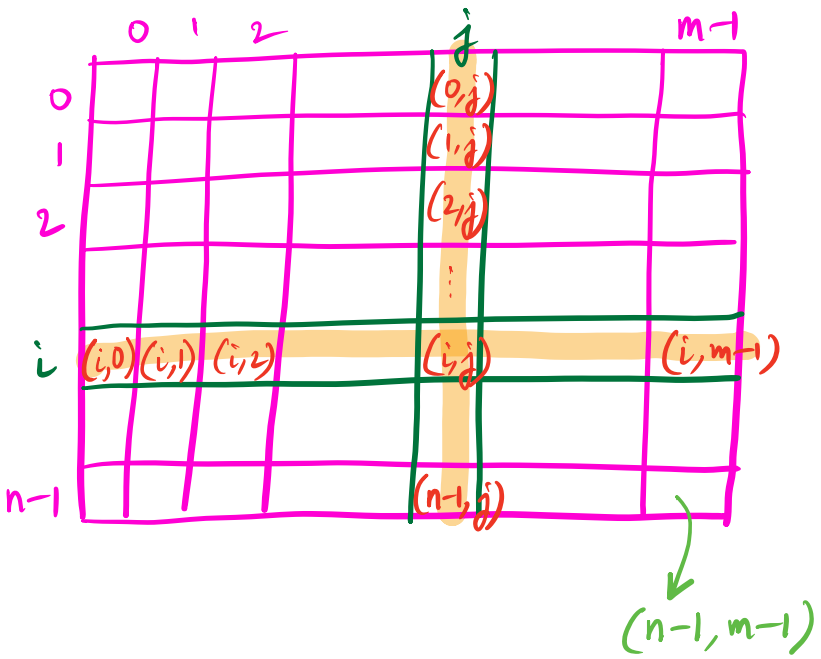
2-D Matrix Basics

```
int mat[][] = new int[4][5]
```

(4 → rows, 5 → columns)



`int mat[n][m]`

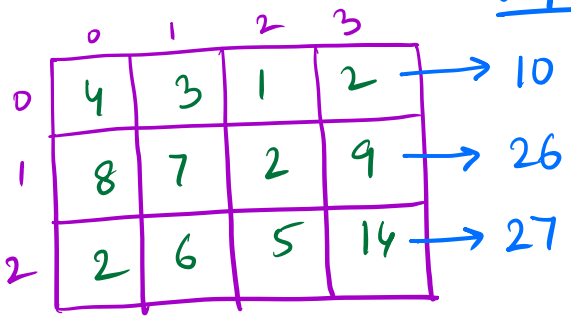


Obs :-

1) If we iterate in a row, (L → R), the col. no. changes from 0 to m-1

2) If we iterate in a column, (T → B), the row no. changes from 0 to n-1

Q1) given mat[n][m], print row wise sum



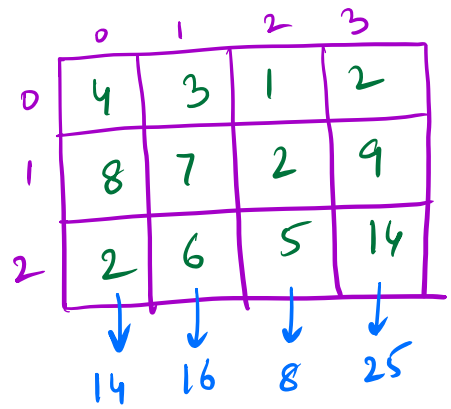
Output

```
void row_sum(int mat[][]){
    int n = mat.length;
    int m = mat[0].length;
    // visit every row
    for(i=0; i<n; i++) {
        int sum = 0;
        // find sum for ith row
        for(j=0; j<m; j++){
            sum += mat[i][j];
        }
        print(sum)
    }
```

T.C. → O(n*m)
S.C. → O(1)

We can't do better.

Q2) given mat[n][m], print column wise sum

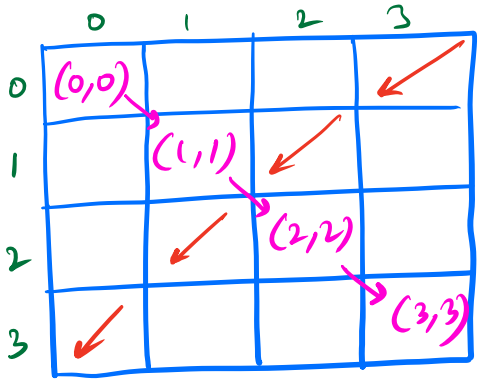


H/W Todo.

Q3) Given square matrix mat[n][n], print two diagonals

1) Top left to Bottom right

2) Top right to Bottom Left

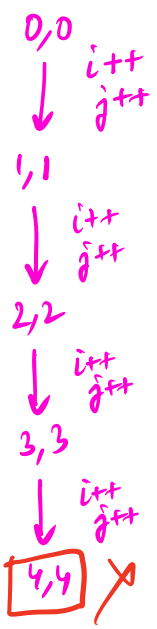


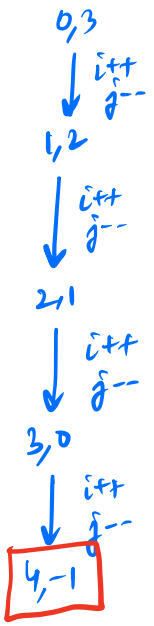

```
i=0, j=0;
while(i<n && j<n){
    print(mat[i][j]);
    i++;
    j++;
}
```

O(n) T.C.
O(1) S.C.

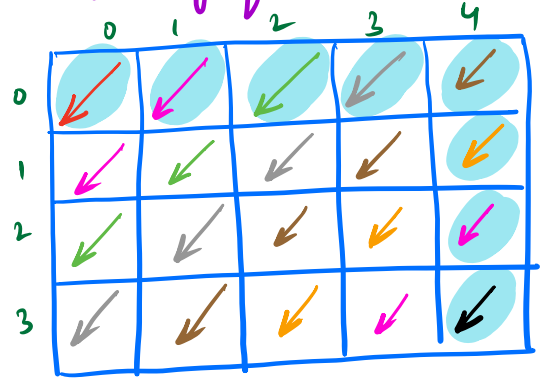
```
i=0, j=n-1;
while(i<n && j>=0){
    print(mat[i][j]);
    i++;
    j--;
}
```

O(n) T.C.
O(1) S.C.





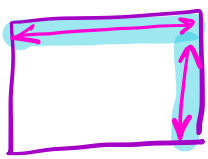
Q4) Given a mat[n][m], print all diagonals going from R→L, starting from 0th row, 0th column.




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

starting points

(0,0) (1,4)
(0,1) (2,4)
(0,2) (3,4)
(0,3)
(0,4)



(0,j) for every $j \in [0, m-1]$

(i, m-1) for every $i \in [1, n-1]$

→ First print all diagonals starting at 0th row

$(0,0) \to (0,1) \to (0,2) \to \dots \to (0,m-1)$

→ Print all the diagonals starting at $(m-1)^{th}$ column.

```
void printAllDiagonals(int mat[][], n, m) {
    for(int c=0; c<m; c++){
        int i=0 , j=c ;
        while(i<n && j>=0) {
            print(mat[i][j]);
            i++;
            j--;
        }
    }
    for(int r=1; r<n ; r++){
        int i=r , j=m-1;
        while(i<n && j>=0) {
            print(mat[i][j]);
            i++;
            j--;
        }
    }
}
```

O(nm) ← first for loop
≤ n times ← while loop

O(nm) ← second for loop
≤ m times ← while loop

O(nm) T.C.

O(1) S.C.

[Break till 11:00 PM]

Q5) Given a matrix mat[n][n], convert it to its transpose.

[O(1) space]

	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

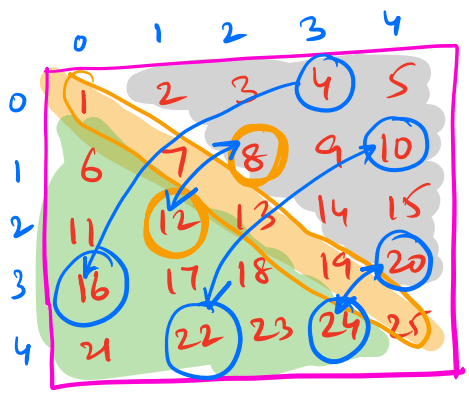
A

→

	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

A^T

$$A[i][j] = A^T[j][i]$$



	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

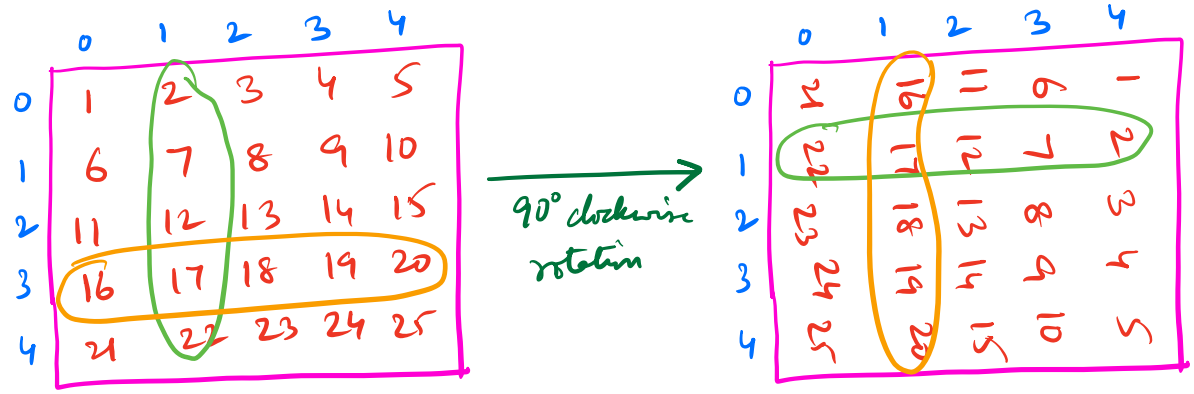
Swap (i,j) with (j,i)

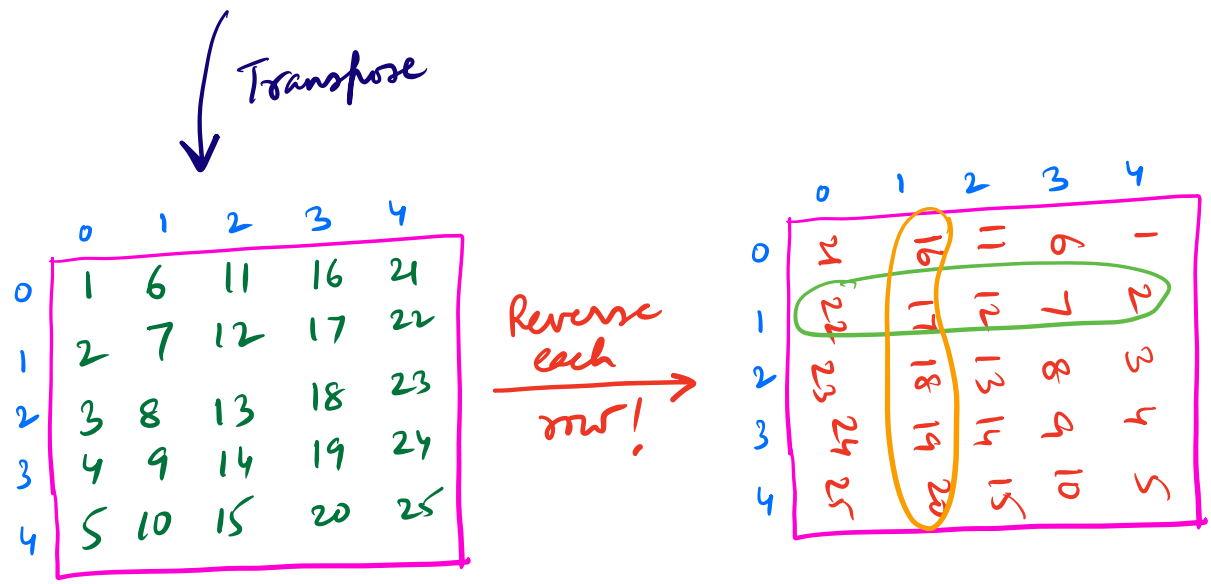
```
for(i=0; i<n; i++){
    for(j=0; j<i; j++){
        int tmp = mat[i][j];
        mat[i][j] = mat[j][i];
        mat[j][i] = tmp;
    }
}
```

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

(0,1) ↔ (1,0) } no net change.
(1,0) ↔ (0,1) }

Q6) Given a matrix mat[n][n], rotate it 90° clockwise.

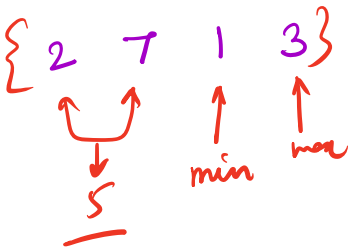




Matrix → Transpose → Reverse every row → 90° clockwise rotated matrix

Todo :- Code

Doubts



2.