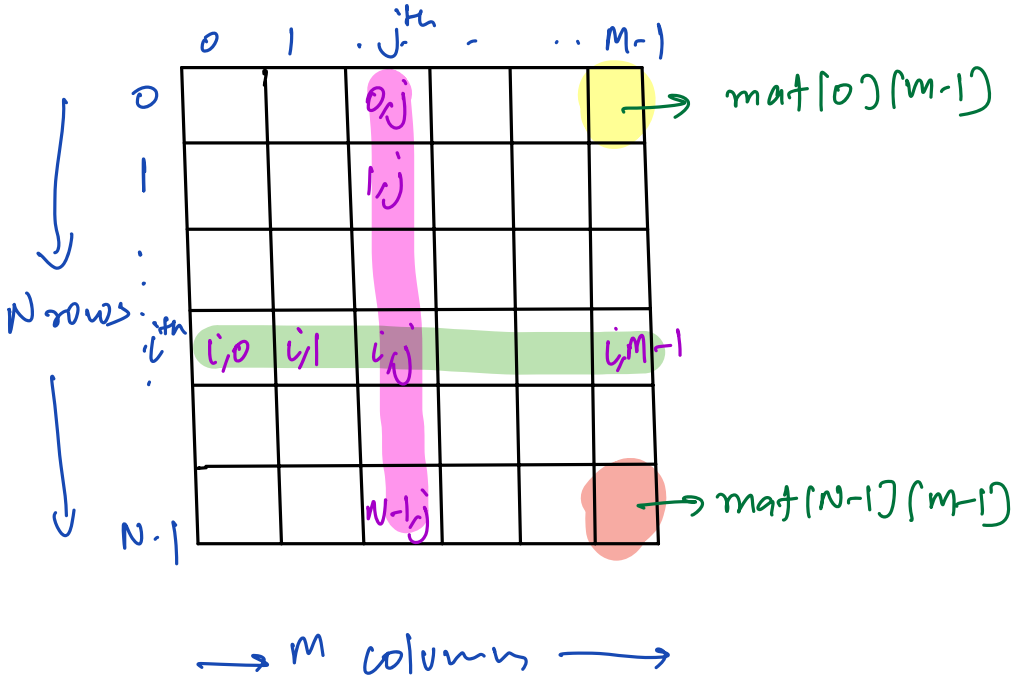
Array: 2D Matrices

2D Matrix → 2D array
it has rectangular grid of values

Declare: int mat[N][M] → learn in your own language

int mat[N][M] → M columns
↳ N rows

Rows are horizontal
Cols are vertical



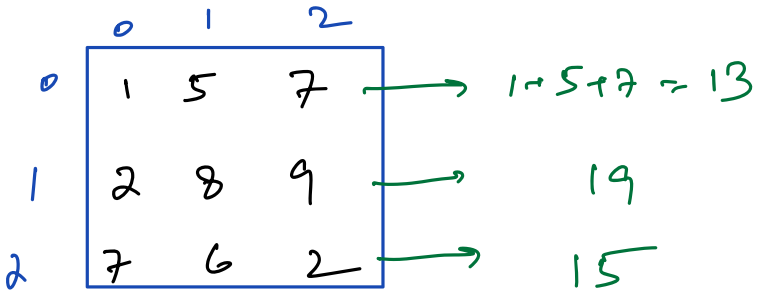
Observation

→ If we move in i^{th} row, row no. is constant
but col. will change → [0, M-1]

→ If we move in j^{th} col, col no. is constant
but row will change → [0, N-1]

Question 1

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




```
function sumRow( mat[N][M] ) {

    for(i=0; i<N; ++i) {
        //ith row
        sum=0;
        for(j=0; j<m; ++j){
            sum += mat[i][j]
        }
        print(sum)

    }
```

SC : O(1)

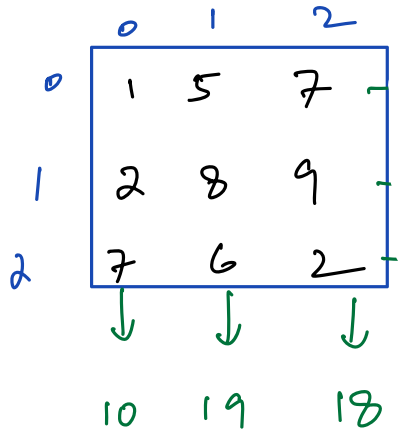
TC : O(N*M)

3

3

Question 2

Given mat[N][M], print column-wise sum.




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

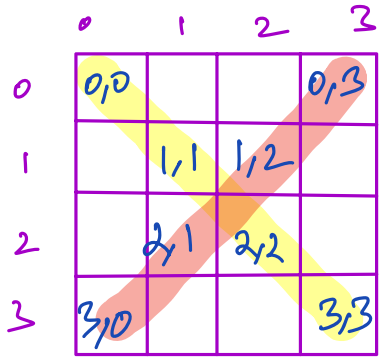
TC: O(N*M)

SC: O(1)

Question 3

Given square mat[N][N], print diagonals.

left → right & right → left



n = 4

0+3 = 3
1+2 = 3
2+1 = 3
3+0 = 3

left to right

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

TC: O(N)
SC: O(1)

OR

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

In right to left

i + j = n-1 ⇒ j = n-1-i

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

TC: O(N)
SC: O(1)

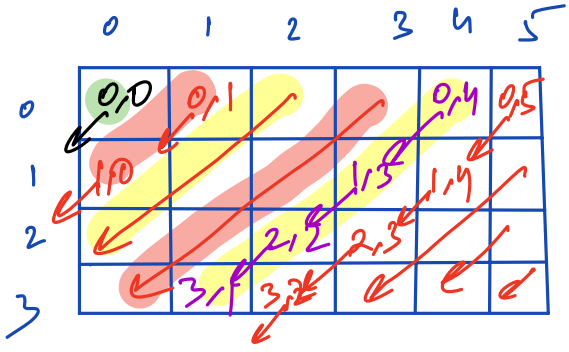
OR

```
for(i=0; i<n; ++i){
    print(mat[i][n-1-i])
}
```

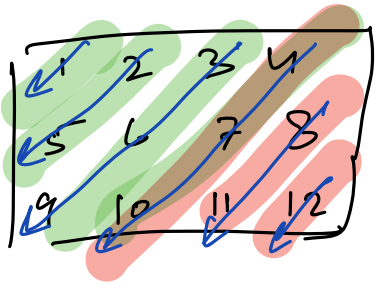
Question 4

Given mat[N][M], print all diagonals, going from right → left.

Diagonals should start from 0th row OR m-1th column.



mat[4][6]



= M + N - 1

= N + M - 1

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

```
printDiagonals(mat[n][m]) {
    // print all diagonals from 0th row

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

println()
}

// print all diagonals from m-1th column

for(i=1; i<n; ++i) {

r = i, c = m-1

while(r<n && c>=0) {

print(mat[r][c])

r++, c--

}

println()

}

TC: O(N+M)

SC: O(1)

Question 5

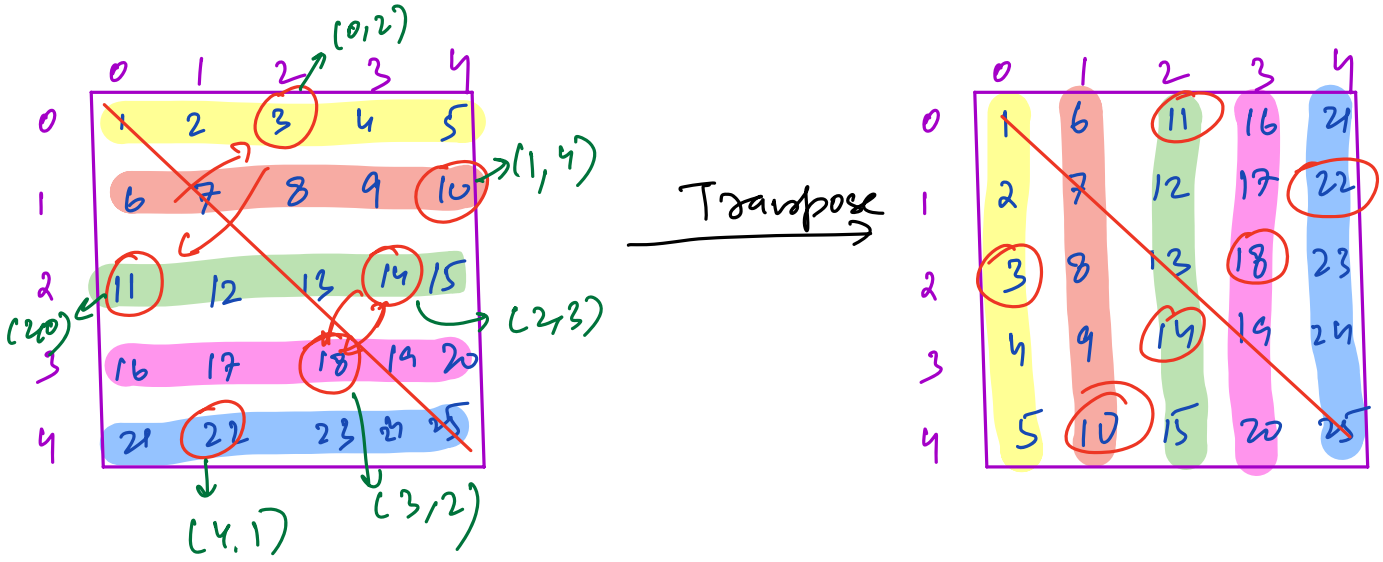
Given mat[N][N], calculate transpose of the matrix w/o extra space.

Transpose:

0th row → 0th col

1st row → 1st col

⋮



$[i,j] \rightarrow [j,i]$

$[i,i] \rightarrow [i,i]$

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

i=0, j=1

[0,1] ⟷ [1,0]

i=1, j=0

[1,0] ⟷ [0,1]

this will do double-swap

& we get original matrix

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

$\Rightarrow i < j$

TC: $O(N^2)$

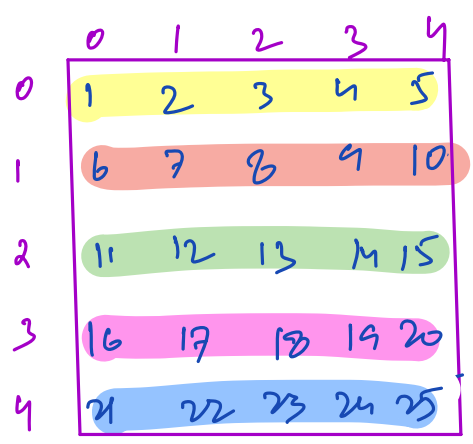
SC: $O(1)$

```
swap(int x, int y) {
    temp = x
    x = y
    y = temp
}
```

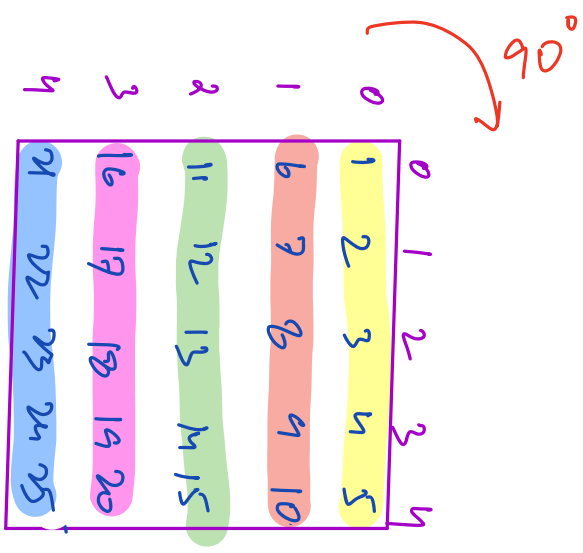
Question 6

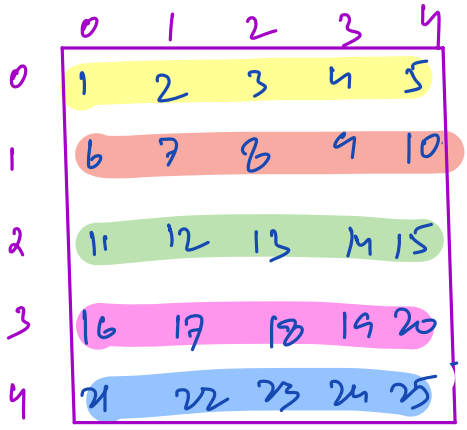
Given mat[N][N], rotate 90° clockwise, from top-right.

SC: $O(1)$

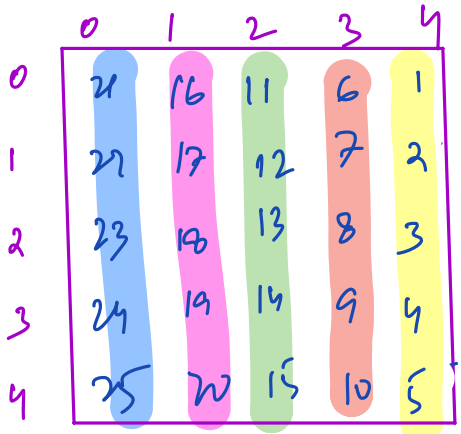


rotate 90° clockwise

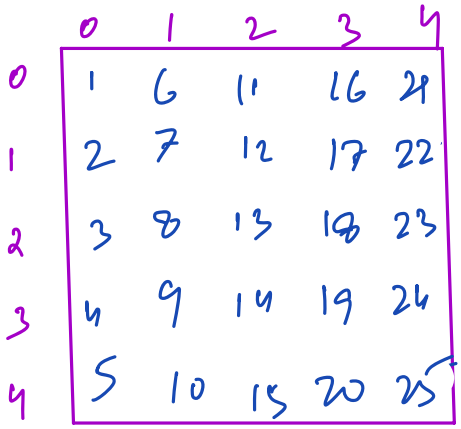




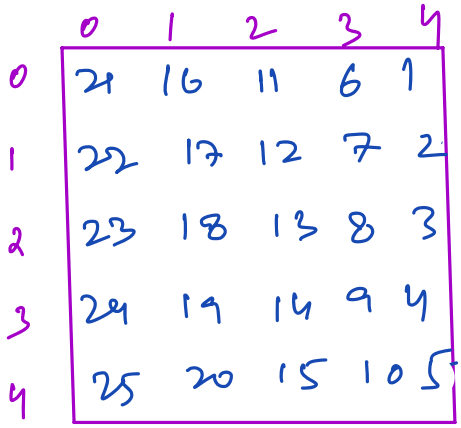
90° clockwise



transpose



reverse



90° clockwise = transpose + reverse

TC: $N^2 + N^2$

$= O(N^2)$

SC: O(1)

Doubt

int a[n]

$a[i] = 10^7$

$n = 10^5$

$\sum a[i] = 10^{12}$

```
sum = 0
for(i=0 to n)
    sum = (sum + a[i]) % (10^9+7)

return sum % (10^9+7)
```

optional