2D Arrays (or) Multidimensional

0	1	2	3	4
50	40	70	90	20

→ marks (maths)

science

eng

	0	1	2	3	4
math 0	50	40	70	90	20
science 1	30	65	50	---	--
eng 2	20	10	50	80	90

Maths
English
science } → marks

st 1 → maths → 50

st 2 → 40, 65, 10

mandatory

int[] marks = new int [5]; →

int[][] marks = new int [3][5]; → new int [ ][5]; X

(2)

rowSize → mandatory

colsize → optional

marks = new int [3][ ]; → empty

X

__ null

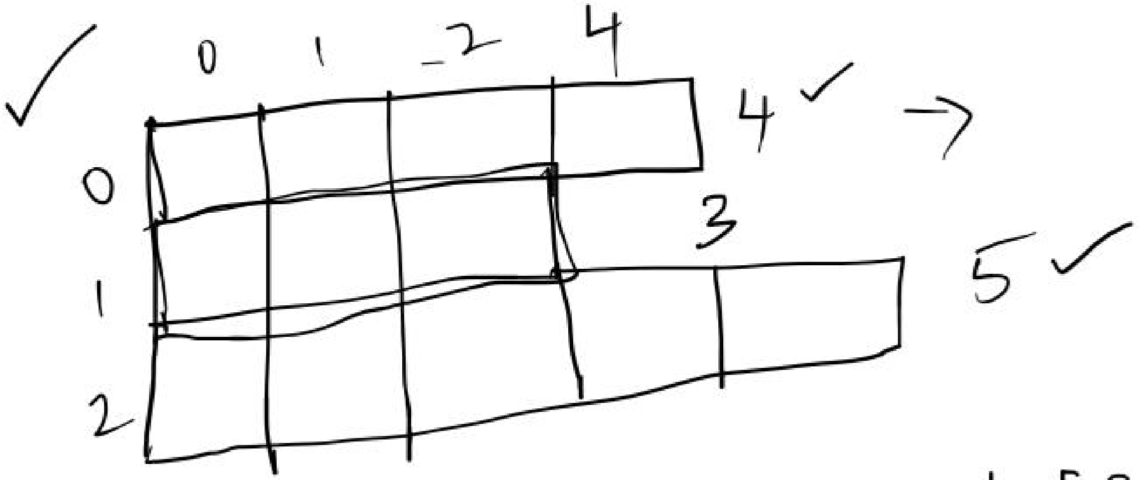
__ null

__ null

Length → 1D array → marks. length → 5

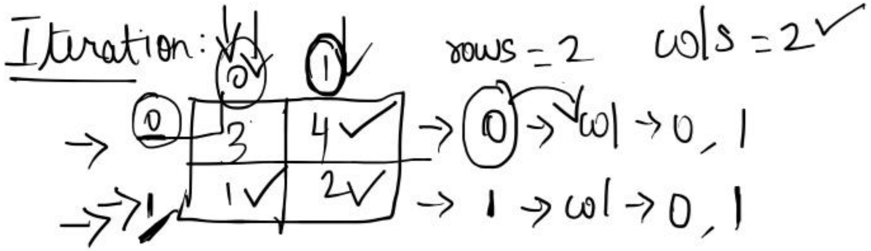
→ 2D array → ✓row length → marks . length → row size

✓Collength → marks[1] . length



```
int [][] marks = new int [3][✓];    null →
                                    null
                                    null
✓marks[1]. = new int [3];
marks [0] = new int [4]
marks [2] = new int [5];
```

Iteration:



```
int[][] nums = new int[2][2];
for(int row = 0; row < nums.length; (row++)) {   → rowsize
    for(int col=0; col < nums[row].length; col++) {
int value = nums[row][col];
    }
}
```

row = 0
    col = 0
    col = 1
    2
row = 1
    col = 0
    col = 1

O/P: 3,4.
1,2

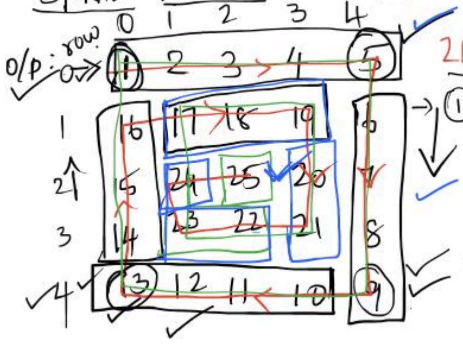
rowlength → nums.length
collength → nums[rowindex].length

nums[0][1] = 3;

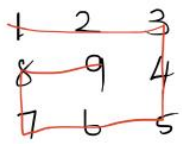
$$\text{int[][][] nums = new int}\overbrace{\text{[][][]}}^{\text{size}}\text{;}$$

int → char, boolean, etc

SPIRAL MATRIX : i/p: n = 5



n= 3



n=1

1

n=0

[[ ]]

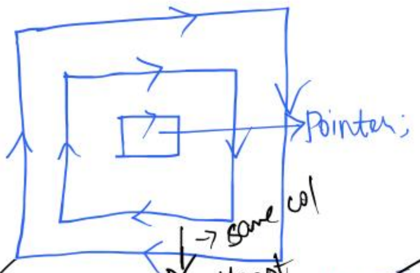
start =1 -> start++;

row ⇒ constant
↓
0 -> row++ (1)
col -> 0 to length -> } -> ①

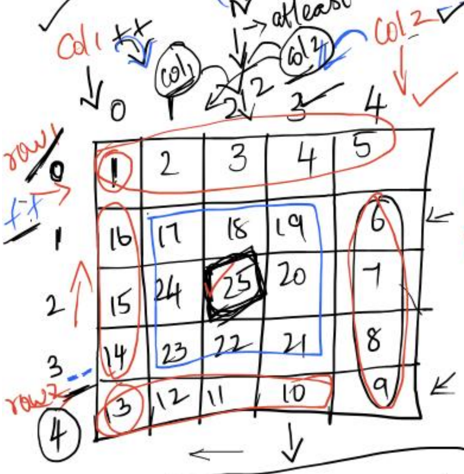
col--
col -> constant (last col)
row -> 1 to last row } -> ②

row--
row -> constant (last row)
col -> length -1 to 0 } -> ③

col -> leng-1 to 1
row -> leng-1 to 1
col -> constant (0) } -> ④



row1=2 row2=2 col1=2, col2=2



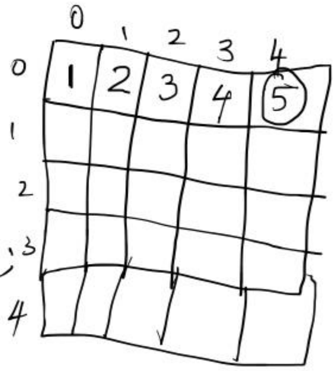
int start = 6 -> incrementu

int row1 = 0; int row2 = n-1;

int col1 = 0; int col2 = n-1;

int[][] matrix = new int[n][n];

while (condition) {



layer1

true

condition: atleast 1 row/col

while (col1 <= col2 && row1 <= row2)

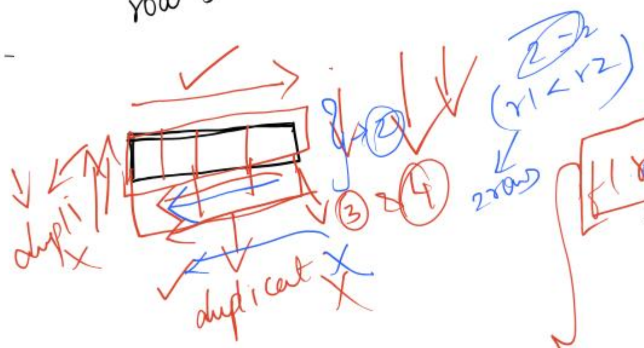
Step1:

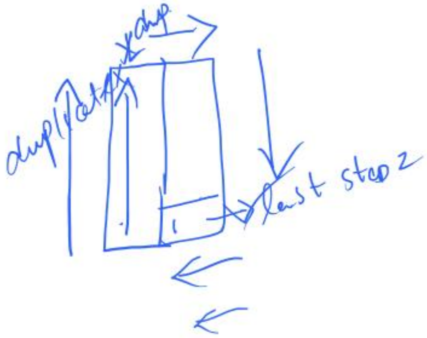
```
① for(int C = col1; C<= col2; C++) {
      matrix[row1][C] = start++; ->
  }
```

C = 5

n=5

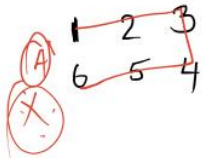
row 5   col=6





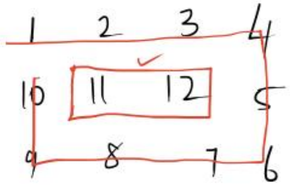
```
}
Step 2:
  for(int r = row1+1; r <= row2 ; r++ {
     matrix[r][col2] = start ++;
  }
if( row1 < row2  && col1 < col2) {
  Step 3: ->
    for(int c =col2 -1 ; c >= col1 ; c --) {
      matrix[row2] [c]  = start++;

    }
  Step 4; ->
    for( int r = row2-1 ; r >= row1+1 , r --) {
      matrix [r] [col1] = start++;

    }
}
row1++;
row2 --;
col1 ++;
col2 --;
}
```

eg: row=2 col=3



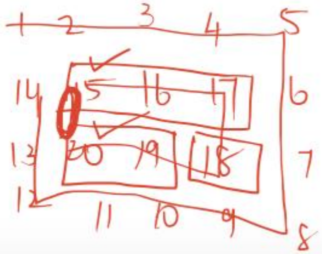
1	2	3
6	5	4

row = 3 col=4



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

row = 4 col = 5

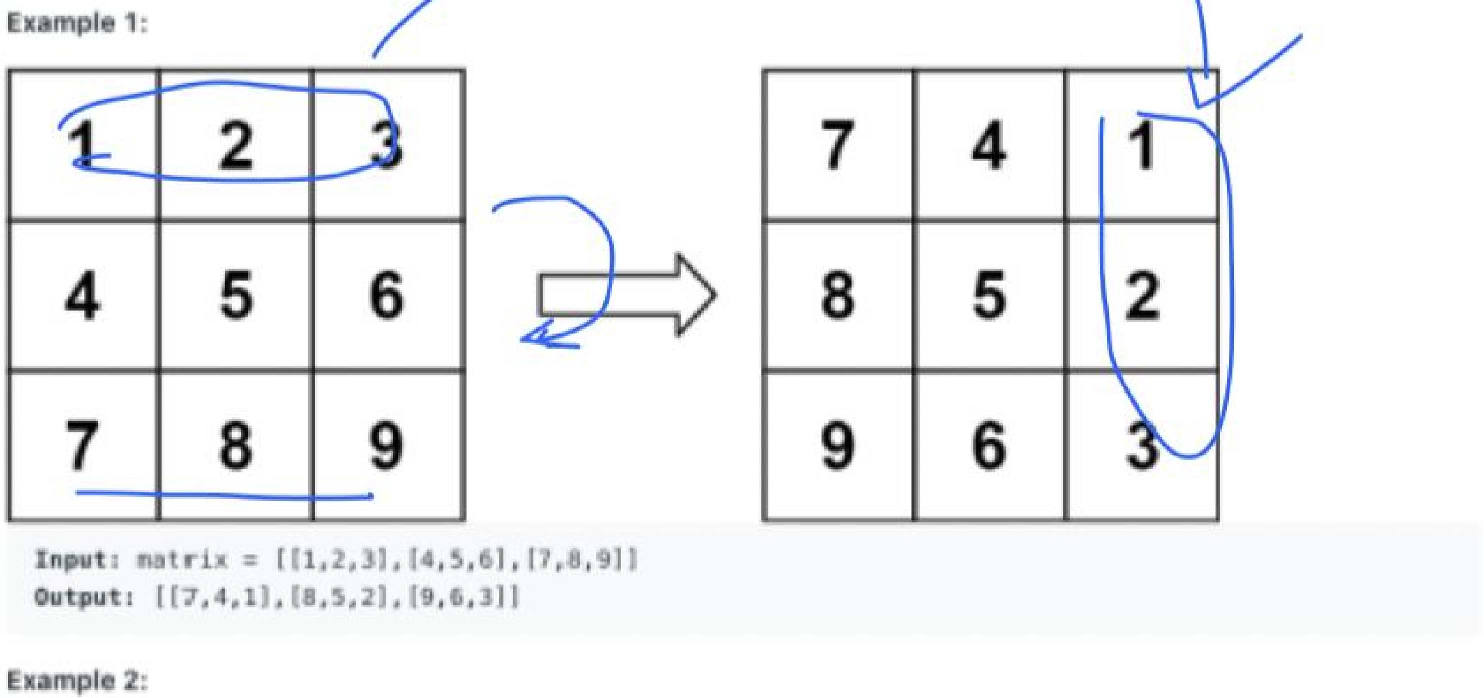


1	2	3	4	5
14	15	16	17	6
13	20	19	18	7
12	11	10	9	8

Problems to work out

Rotate Matrix

7
4
9



Example 1:

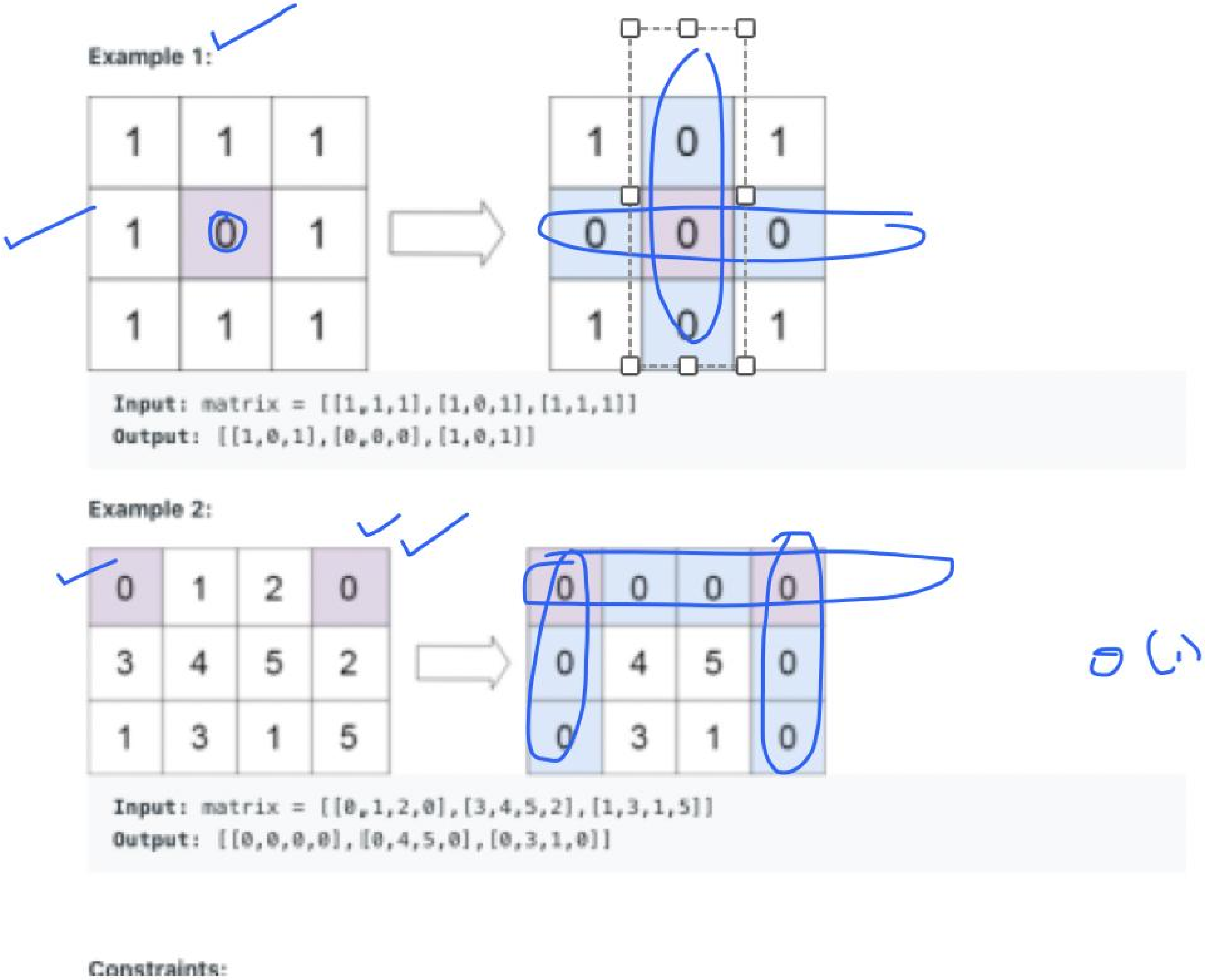
1	2	3
4	5	6
7	8	9

7	4	1
8	5	2
9	6	3

```
Input: matrix = [[1,2,3],[4,5,6],[7,8,9]]
Output: [[7,4,1],[8,5,2],[9,6,3]]
```

Example 2:

Set matrix Zeroes



Example 1:

1	1	1
1	0	1
1	1	1

1	0	1
0	0	0
1	0	1

```
Input: matrix = [[1,1,1],[1,0,1],[1,1,1]]
Output: [[1,0,1],[0,0,0],[1,0,1]]
```

Example 2:

0	1	2	0
3	4	5	2
1	3	1	5

0	0	0	0
0	4	5	0
0	3	1	0

O(1)

```
Input: matrix = [[0,1,2,0],[3,4,5,2],[1,3,1,5]]
Output: [[0,0,0,0],[0,4,5,0],[0,3,1,0]]
```

Constraints: