Todays Content

1. Transpose
2. Rotate 90°
3. Spiral printing
4. Matrix Multiplicatin
5. Make zero

Mat as vector:

Vector< Vector<int> > v;

v:

	0	1	2	3	4
v[0] = {	3	5	30	7	9 }
v[1] = {	1	2	4	7	50 }
v[2] = {	10	9	6	11	12 }
v[3] = {	14	7	6	7	9 }

v.size() = 4

v[0].size() = 5

Note: Mat[0][0] → vector<vector<int>> v;

v.size() # No: of rows

v[0].size() # No: of columns

v[0][2] = 30;

v[1][4] = 50;

Q2: Given mat[N][N] print boundary in clockwise direction, start from (0,0)

Constraints:

$1 \le N * N \le 10^6$

$-10^6 \le mat[i][j] \le 10^6$

tc1: Mat[5][5]

	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

Output: 1 2 3 4 5 10 15 20 25 24 23 22 21 16 11 6

tc2: mat[3][3]

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

Output: 1 2 3 6 9 8 7 4

Ideas: Mat[5][5]

	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

Steps:

1. Iterate 4 times print 0th row L→R
2. Iterate 4 times print last col T→D
3. Iterate 4 times print last row R→L
4. Iterate 4 times print 0th col D→T

Mat[6][6]

	0	1	2	3	4	5
0	1	2	3	4	5	6
1	7	8	9	10	11	12
2	13	14	15	16	17	18
3	19	20	21	22	23	24
4	25	26	27	28	29	30
5	31	32	33	34	35	36

Steps:

1. Iterate 5 times print 0th row L→R
2. Iterate 5 times print last col T→D
3. Iterate 5 times print last row R→L
4. Iterate 5 times print 0th col D→T

Gen; mat[N][N]: iterate N-1 times.

```
void printBoundary(int[][] mat) {
    int N = mat.size();
    int i=0, j=0;
    #Step1 : Iterate N-1 times print 0th row L→R
    for(int l=1; l<N; l++) { //l=[1..N-1]
        print(mat[i][j]);
    3   j++; // L→R
    #Step2 Iterate N-1 times print last T→D
    for(int l=1; l<N; l++) { //l=[1..N-1]
        print(mat[i][j]);
    3   i++; // T→D
    #Step3 Iterate N-1 times print last row R→L
    for(int l=1; l<N; l++) { //l=[1..N-1]
        print(mat[i][j]);
    3   j--; // R→L
    #Step Iterate N-1 times print 0th col D→T
    for(int l=1; l<N; l++) { //l=[1..N-1]
        print(mat[i][j]);
    3   i--; // D→T

    if(N==1){ print(mat[i][j]); }
3
```

i=0, j=0, N=5

l	l<N	mat[i][j]	j++ (i, j)
1	1<5	mat[0][0] ✓	(0, 1)
2	2<5	mat[0][1] ✓	(0, 2)
3	3<5	mat[0][2] ✓	(0, 3)
4	4<5	mat[0][3] ✓	(0, 4)
5	5<5 : Stop		

i=0, j=4

l	l<N	mat[i][j]	i++ (i, j)
1	1<5	mat[0][4] ✓	(1 4)
2	2<5	mat[1][4] ✓	(2 4)
3	3<5	mat[2][4] ✓	(3 4)
4	4<5	mat[3][4] ✓	(4 4)
5	5<5 : Stop		

i=4, j=4

l	l<N	mat[i][j]	j-- (i, j)
1	1<5	mat[4][4] ✓	(4 3)
2	2<5	mat[4][3] ✓	(4 2)
3	3<5	mat[4][2] ✓	(4 1)
4	4<5	mat[4][1] ✓	(4 0)
5	5<5 : Stop		

i=4, j=0

l	l<N	mat[i][j]	i-- (i, j)
1	1<5	mat[4][0] ✓	(3, 0)
2	2<5	mat[3][0] ✓	(2, 0)
3	3<5	mat[2][0] ✓	(1, 0)
4	4<5	mat[1][0] ✓	(0, 0)
5	5<5		

Edge case

N=1, mat[1][1] = 0

0
10

expected output: 10

obs: After completing boundary code, we will go back to start point.

Spiral Printing

Mat[6][6]:

↘	0	1	2	3	4	5
0	1	2	3	4	5	6
1	7	8	9	10	11	12
2	13	14	15	16	17	18
3	19	20	21	22	23	24
4	25	26	27	28	29	30
5	31	32	33	34	35	36

i	j	N : N-1 iterating	i	j	i++	j++	N == 2
0	0	6	5 iterating	0, 0	1	1	4
1	1	4	3 iterating	1, 1	2	2	2
2	2	2	1 iterating	2, 2	3	3	0 ⇒ Stop

↘	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

i	j	N : N-1 iterating	i	j	i++	j++	N == 2
0	0	5 : 4 iterating	0	0	1,	1	3
1	1	3 : 2 iterating	1	1	2	2	1 : Stop
2	2	1 : 0 iterating : Cannot print.					

if (N == 1) : print (mat[i][j]); # Centre element

```
void SpiralPrinting(vector<vector<int>> mat){ TC: O(N^2) SC: O(1)
    int N = mat.size();
    int i=0, j=0;
    while(N>1){
        #Step1 : Iterate N-1 times print 0th row L->R
        for(int k=1; k<N; k++){ //k=[1..N-1]
            print(mat[i][j]);
        }  j++; // L->R
        #Step2 Iterate N-1 times print last T->D
        for(int k=1; k<N; k++){ //k=[1..N-1]
            print(mat[i][j]);
        }  i++; // T->D
        #Step3 Iterate N-1 times print last row R->L
        for(int k=1; k<N; k++){ //k=[1..N-1]
            print(mat[i][j]);
        }  j--; // R->L
        #Step4 Iterate N-1 times print 0th col D->T
        for(int k=1; k<N; k++){ //k=[1..N-1]
            print(mat[i][j]);
        }  i--; // D->T
        i++; j++; N=N-2;
    }
    if(N==1){ print(mat[i][j]); }
}
```

38 Transpose of mat[N][N] without extra space

Transpose

0^{th} row ⟶ 0^{th} col

1^{st} row ⟶ 1^{st} col

2^{nd} row ⟶ 2^{nd} col

⋮

$N-1^{st}$ row ⟶ $N-1^{st}$ col

mat[5][5]

↘	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

#obs =

↘	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

```
void transpose(vector<vector<int>> mat){

}
```

eg: mat[3][3]

i \ j	0	1	2
0	10	20	30
1	40	50	60
2	70	80	90

Tracing

i = 0, j = 0 :

j = 1 : _____

j : 2

i = 1, j = 0 : _____

#Handle Issues:

```
void transpose(vector<vector<int>> mat){

}
```

	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