Today's Agenda :-

Sunday Problem Solving Session

optional ← ✓ 1-3 pm → least solved assignment 1 h.w questions

or

5-7 pm.

→ 2d matrices

Print row wise

Print max column sum

Print mat[N][N] diagonals

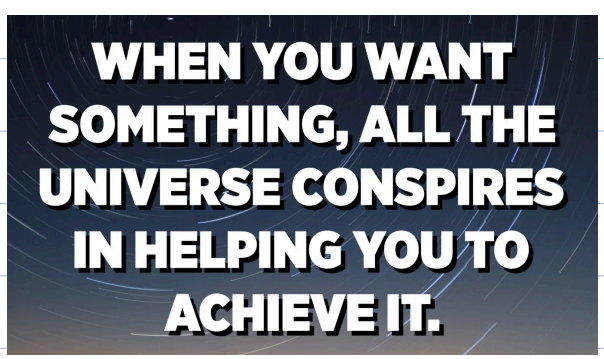
Print mat[N][M], R-L diagonals

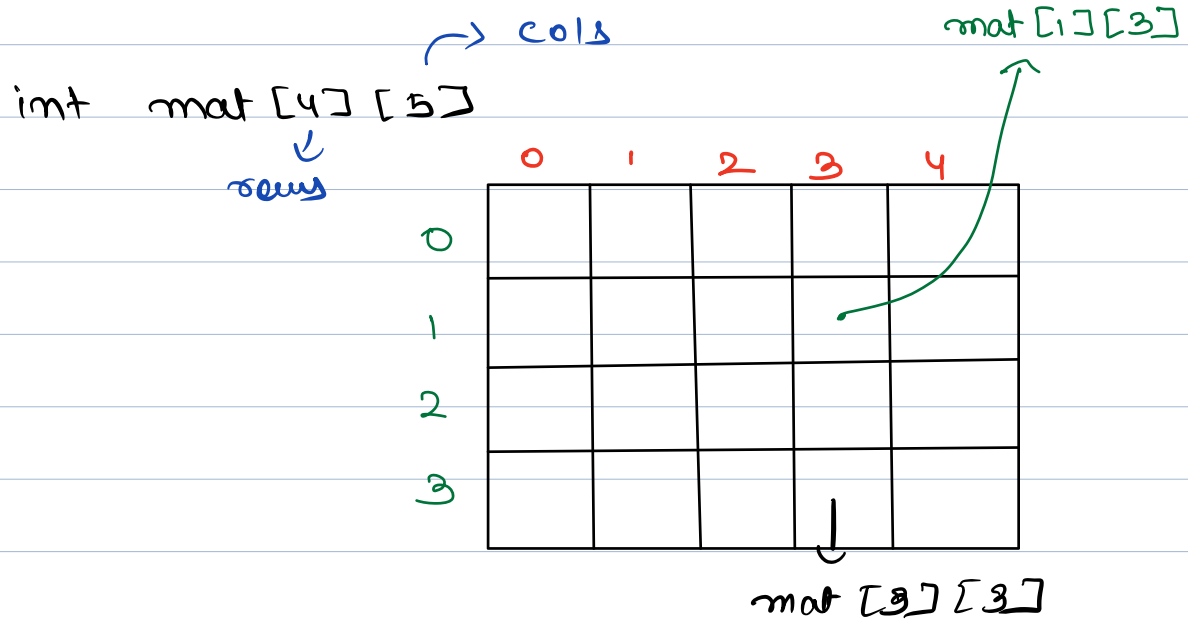
Transpose

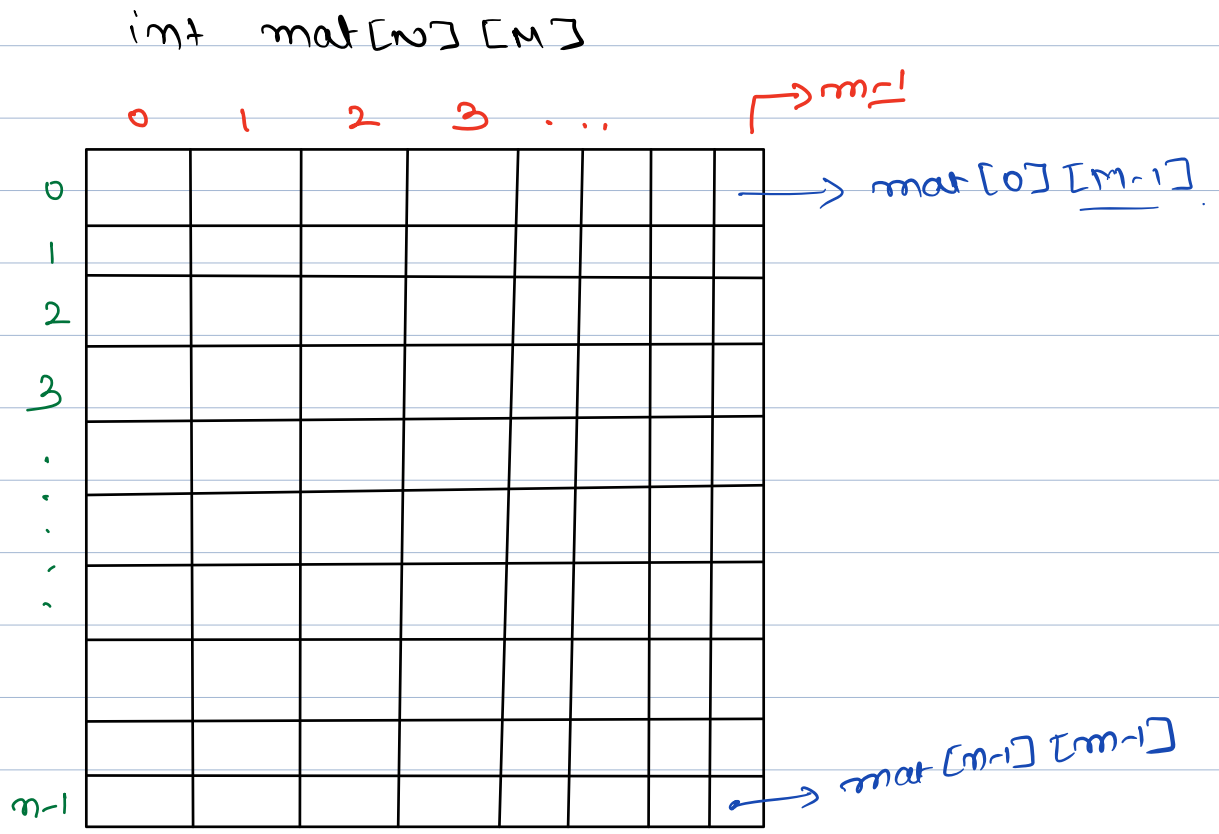
Rotate 90° clockwise

Print all boundaries in clockwise

Quote :-

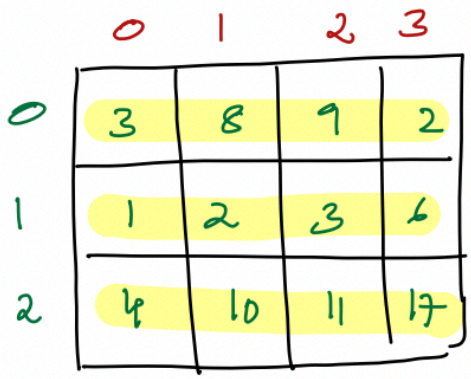



int mat [4] [5]
rows
cols
0 1 2 3 4
0
1
2
3
mat [1] [3]
mat [3] [3]


int mat [N] [M]
0 1 2 3 . . .
m-1
0
1
2
3
n-1
mat [0] [M-1]
mat [n-1] [m-1]

1Q) Given mat[N][M], print row-wise →

mat[3][4]




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

3 3

3

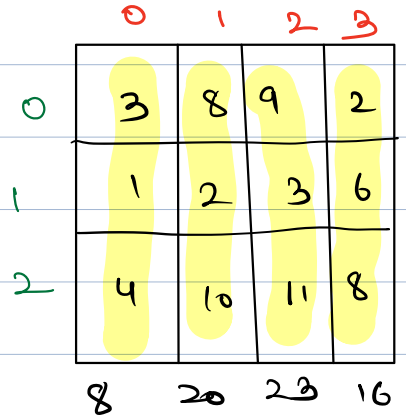
i	J
0	0
0	1
0	2
0	3
1	0
1	1
1	2
1	3

output → 3 8 9 2
1 2 3 6
4 10 11 17

T.C → O(n*m)

S.C → O(1)

2Q) Given mat[N][M], find max column sum →



T.C -> O(N*M)

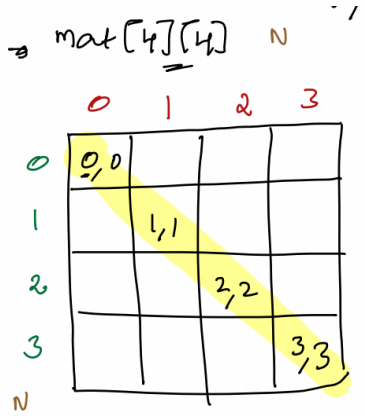
S.C -> O(1)

Ans -> 23

```
maxSum = Integer.MIN_VALUE;
for (j=0; j<M ; j++) {
    sum=0
    for (i=0; i<N; i++) {
        sum+ = mat[i][j]
    }
    maxSum= Max (maxSum, sum)
}
```

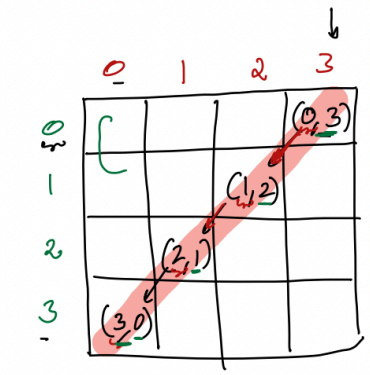
j	i
0	0
0	1
0	2

Ques 3) Given a mat[N][N], print diagonals → left to right
→ right to left.




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

T.C → O(n)

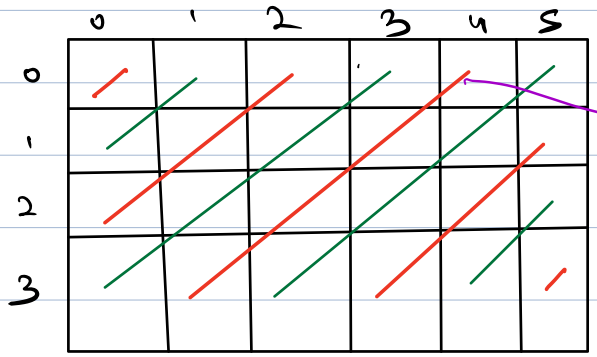



```
i=0, j=n-1
while (i<N&&j>=0) {
    print(mat[i][j])
    i++, j--
}
```

T.C → O(n)
S.C → O(1)

4Q) Given mat[N][M] print all diagonals going R-L

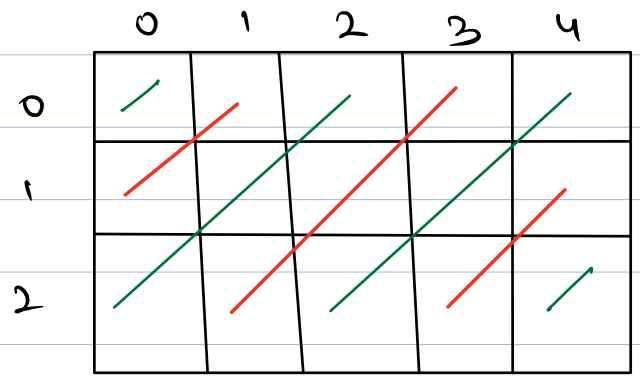
mat[4][6] , diagonal = 9



no. of diagonals = N + M - 1

(0,4)
↓
(1,3)
↓
(2,2)
↓
(3,1)
↓
(4,0)
↑
row ended.

mat[3][5] , diagonals = 7 .



for (J = 0; J < M; J++)

```
(x,y) = 0,3
while (x<N & y>=0 )
    print (mat[x][y])
    x++, y--
}
```

```
for (i = 1; i < N; i++) {
    (x,y) = (i, M-1)
    while (x<N & y>=0 )
        print (mat[x][y])
        x++, y--
    }
}
```

T.C → O(M * N)

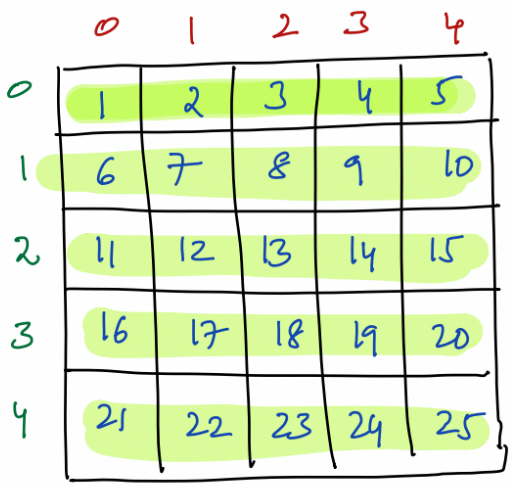
S.C → O(1)

10:10 → 10:18 pm.

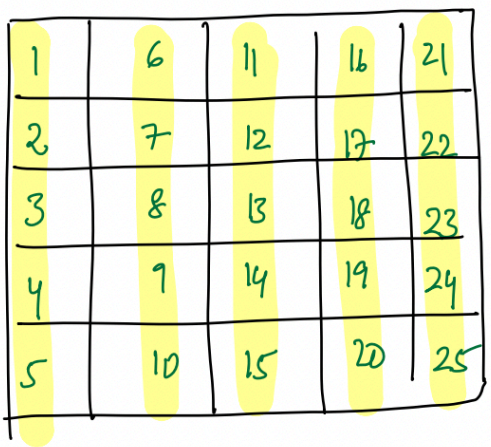
5Q) Given a mat[N][N] find the transpose inplace

1) Given input mat[][] should update, SC: O(1)

mat[5][5]

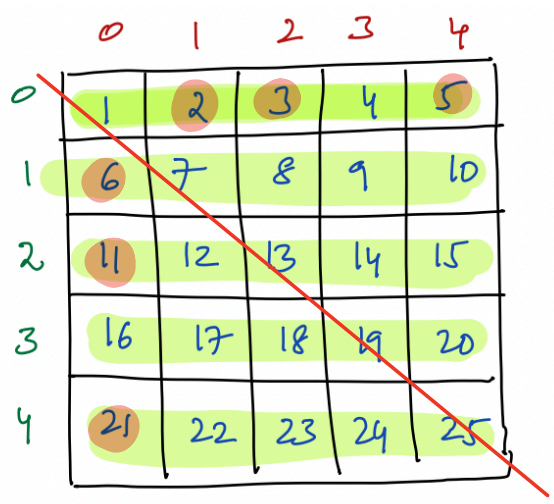


	0	1	2	3	4
0	1	2	3	4	5
1	6	7	8	9	10
2	11	12	13	14	15
3	16	17	18	19	20
4	21	22	23	24	25



1	6	11	16	21
2	7	12	17	22
3	8	13	18	23
4	9	14	19	24
5	10	15	20	25

Approach :-



	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

mat[0][1]

↓↑

mat[1][0]

mat[0][4]

↓↑

mat[4][0]

for (i ———)

```
for (j ___ )
    if (i<j) {
  swap(mat[i][j], mat[j][i])
    }
}
```

3

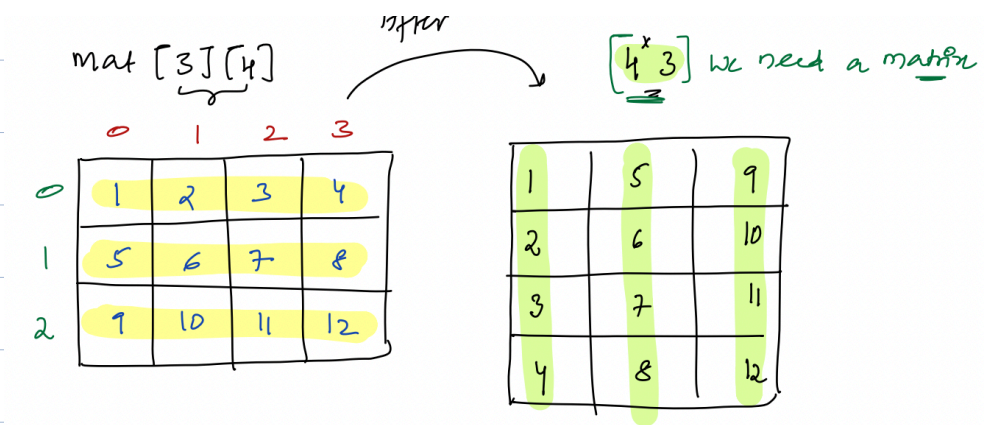
(i,j) → (0,1)

↓

(1,0)

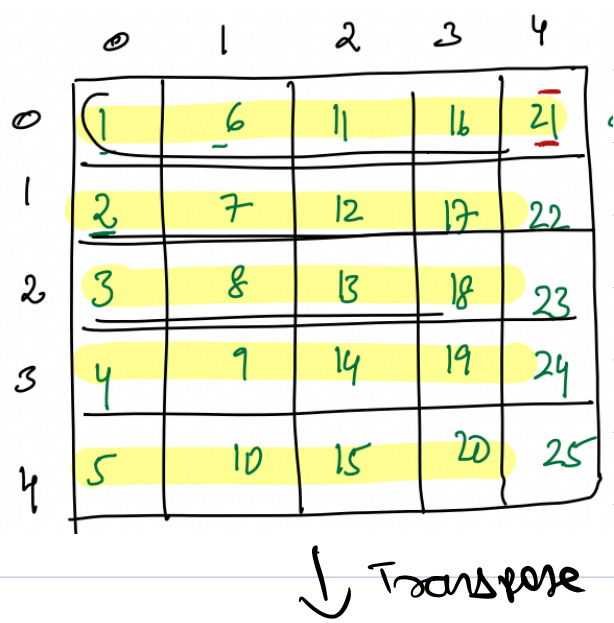
T.C → $O(N^2)$

S.C → $O(1)$

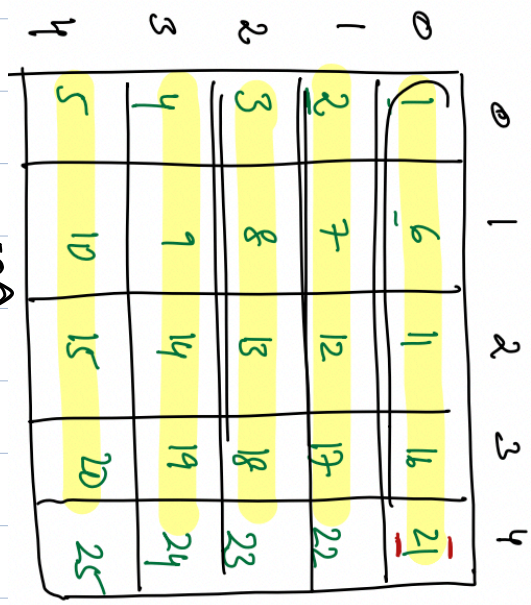


Ques)

GQ) Given a mat[N][N] rotate 90° clockwise, SC: O(1)



90° rotate →



↓ Transpose

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

→ reverse each row

Code :- Todo

reverse ↑

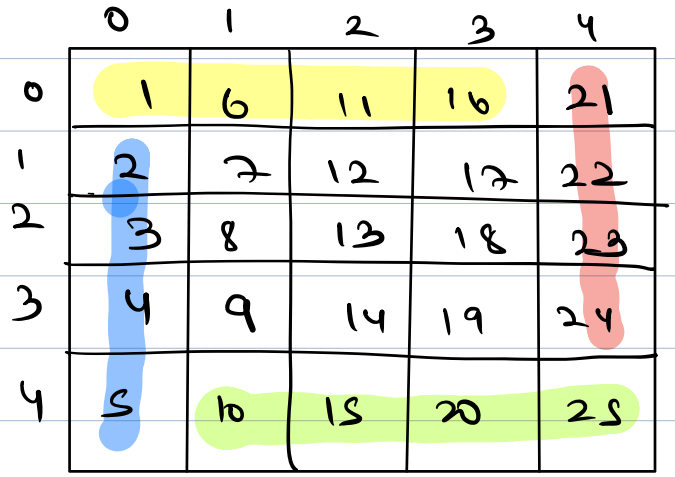
T.C → $N^2 + \frac{N}{2} * N$

$O(N^2)$

S.C → O(1).

Ques)

→ Given a mat[N][N] print all boundaries in clockwise



	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

mat[5][5].

Output

Print 0th row

[0 0] → [0 3] → 4 ele

Print N-1th col :-

[0,4] → [3,4] → 4 ele

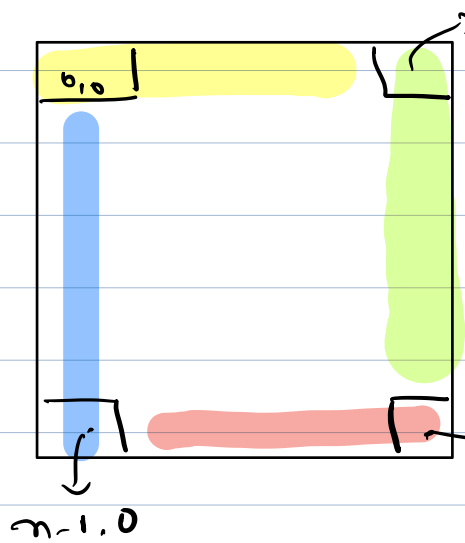
Print N-1th row.

[4 4] → [4,] → 4 ele.

Print 0th col.

[4,0] → [1,0] : 4 Elem.

N * N Matrix



[0 0] → [0, N-2] → N-1 elem.

[0 N-1] → [N-2, N-1] → N-1 ele

[N-1 N-1] → [N-1, 1] → N-1 ele.

[N-1, 0] → [1, 0] → N-1 elem.

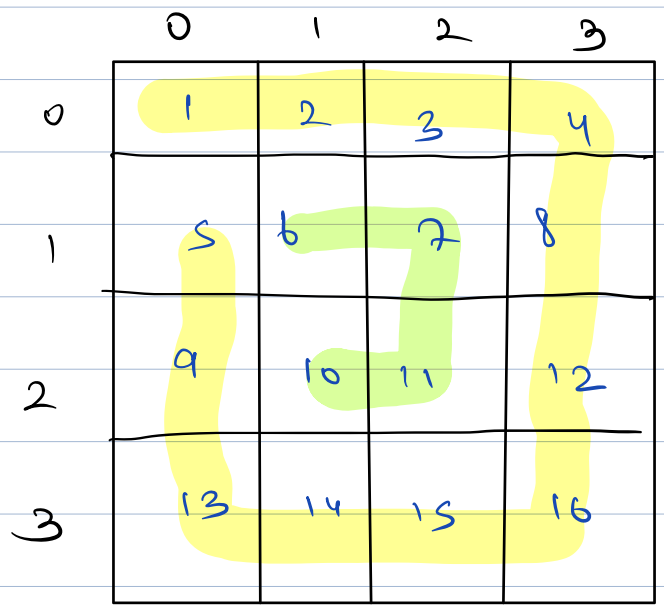
```
i = 0, j = 0

for (k=1; k<N; k++) {
        print (mat[i][j])
        j++
  }

//i=0 , j = N-1
for (k=1; k<N; k++) {
        print (mat[i][j])
        i++
  }

// i = n-1 , j = N-1
for (k=1; k<N; k++) {
        print (mat[i][j])
        j--
```

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

// i=0, j=0

T.C → O(n).

Ques) Given matrix[n][n], print spirally.



→ mat[4][4]

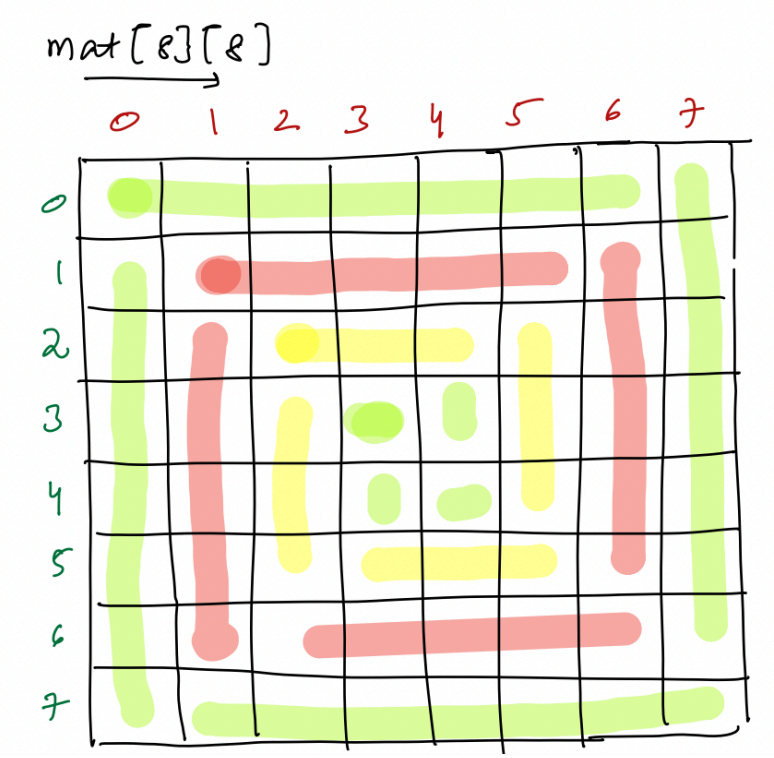
```
i = 0, j = 0
while (N > 0) {
    for (k = 1; k < N; k++) {
        print (mat[i][j])
        j++
    }

    // i = 0, j = N-1
    for (k = 1; k < N; k++) {
        print (mat[i][j])
        i++
    }

    // i = n-1, j = N-1
    for (k = 1; k < N; k++) {
        print (mat[i][j])
        j--
    }

    // i = n-1, j = 0
    for (k = 1; k < N; k++) {
        print (mat[i][j])
        i--
    }

    i++, j++, N -= 2
```

```
if (N == == 1) {
    Print(mat[i][j])
}
```

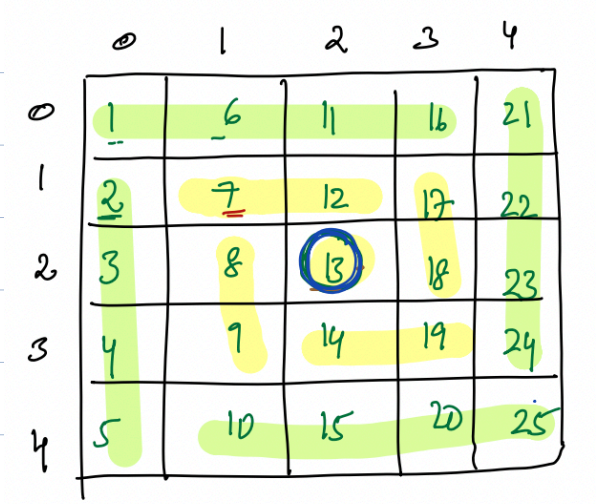
T.C → O(N²)

S.C → O(1)

}



1 edge case :-



(i, j) → 0, 0

N = 5

↓

1, 1

N = 3

↓

2, 2

N = 1

Output

1, 6, 11, 16, 21, 22, 23, 24, 25, 20, 15, 10, 5, 4, 3, 2, 7, 12, 17, 18, 19, 14, 9, 8, 13

Sum += ((long) A[i] * (i+1) * (n-i))

Sum += (long)(A[i] * [i+1][n-i])

0 1 2 3 4

5, -2, 3, 1, 2 S = 3

→