Today's Questions:

① State of Wakanda I { zig-zag order }
② State of Wakanda II { Diagonal order }
③ Spiral Order
④ Exit Point

① State of Wakanda I

	0	1	2	3
0	11	12	13	14
1	21	22	23	24
2	31	32	33	34
3	41	42	43	44

order → 11, 21, 31, 41, 42, 32, 22, 12, 13, 23, 33, 43, 44, 34, 24, 14

Start from column zero upto m-1
  & alternatively traverse from (0 to n-1) or (n-1 to 0)
    print the corresponding element
    update the column

CODE

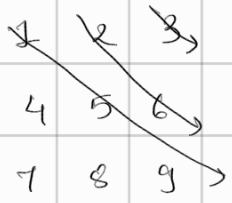
```
void waveTraversal(int[][] mat){
    int rows = mat.length, col = mat[0].length;
    for(int j=0; j<col; j++){
        if(j%2==0)
            for(int i=0; i<rows; i++)
                System.out.println(mat[i][j]);
        else
            for(int i=rows-1; i>=0; i--)
                System.out.println(mat[i][j]);
    }
```

To print matrix column by column

```
for (int j=0; j<m; j++){
    for(int i=0; i<n; i++){
        System.out.print(mat[i][j] + "\t");
    }
    System.out.println("");
}
```

(2) State of Wakanda II

for matrix



traverse as: 1 → 5 → 9 → 2 → 6 → 3.

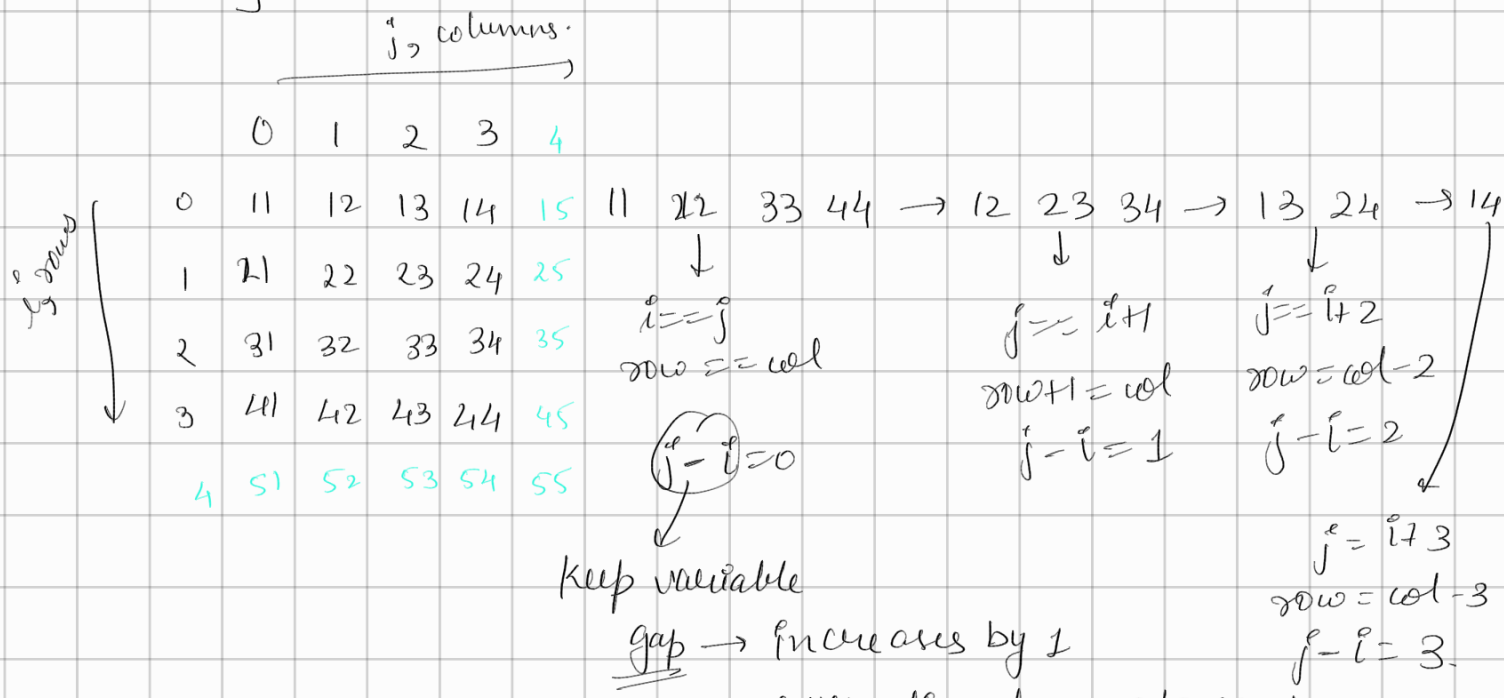
observation, diagonal starts always from i=0 / row=0 and column always increases by 1 for every diagonal start.

after the starting point,
print every element
st row ++, st col ++
upto when col < m.

CODE:

```
int j=0;
```

```
while (j < n) {
    int row = 0, col = j;
    while (col < n) {
        System.out.println(mat[row][col]);
        row++;
        col--;
    }
    j++;
}
```



Keep variable gap → increases by 1 every time from 0 to n-1

or gap can go from 0 to number of columns in rectangle matrix

```
for (gap = 0; gap < n; gap++) {
    for (int i = 0; i + gap < n; i++)
        System.out.println(mat[i][i+gap]);
```

or

```
for (int gap = 0; gap < m; gap++) {   // gap < no. of columns
    // gap = j - i → j = i + gap.
    for (int i = 0; i + gap < n; i++)
        System.out.println(mat[i][i+gap]);
}
```

or

```
for (int gap=0; gap<m; gap++){
    for (int i=0, j=gap; j<m; i++,j++)
        System.out.println(mat[i][j]);
}
```

③ Spiral Order.

Eg:

	0	1	2
0	11	12	13
1	21	22	23
2	31	32	33

11 21 31 32 33 23 13 12 22

top column

Eg:

top row →	0	1	2	3	4	5	6
0	11	12	13	14	15	16	17
1	21	22	23	24	25	26	27
2	31	32	33	34	35	36	37
3	41	42	43	44	45	46	47
4	51	52	53	54	55	56	57 ← bottom row

bottom column.

for outermost spiral

- Left wall : Top to Bottom
- Bottom wall : Left to Right
- Right wall : Bottom to Top
- Top wall : Right to left.

Print corner elements only once

for outermost loop.

Left : loop from i=0 to n-1 print mat[i][0]

Bottom: loop from j=1 to m-1 print mat[n-1][j]

Right : loop from i=n-2 to 0 print mat[i][m-1]

Top : loop from j=m-2 to 1 print mat[0][j]

Logic

Use Row & Column for corners.

∴ Left wall print from first Row to last Row in firstCol

Bottom wall print from firstCol+1 to lastCol in last Row

Right wall print from last Row -1 to first Row in lastCol

Top wall print from lastCol-1 to firstCol+1 in firstRow

update firstRow++, firstCol++, lastCol--, lastRow--;

for middle elements to avoid duplicate elements

always check & return if completed all elements

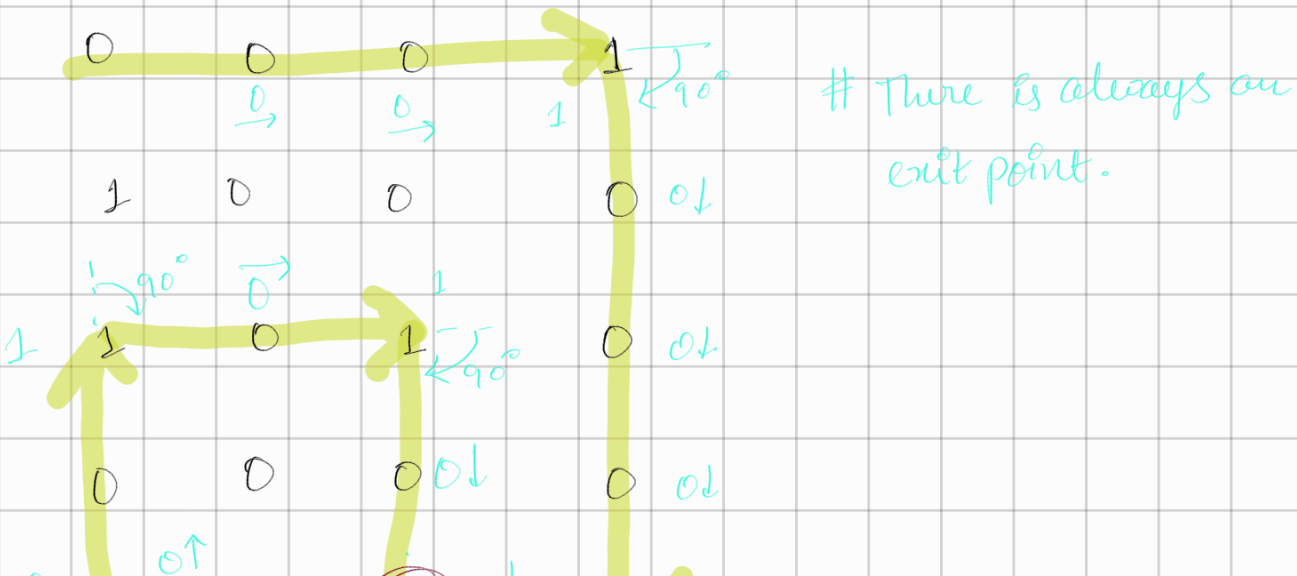
CODE

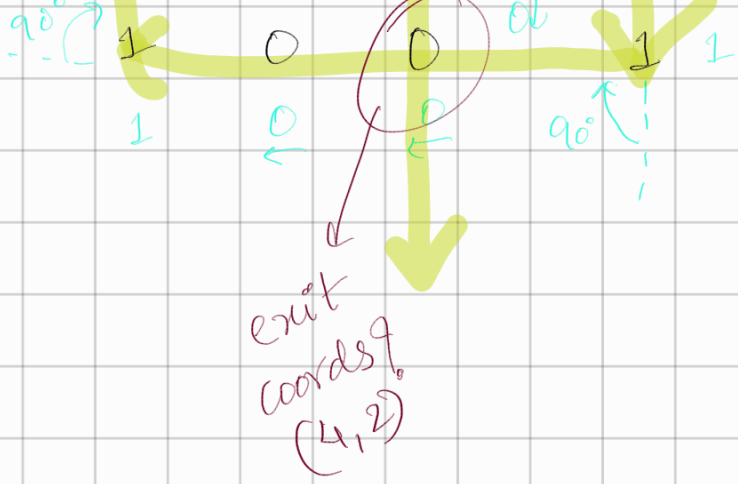
```
int fRow=0, fCol=0, lRow= n-1, lCol= m-1;
int count =0;
while ( count < n*m){
    // left wall from top to bottom
    for(int i= fRow; i<= lRow; i++) {
        System.out.println( mat[i][fCol];
        count++;
        if (count == n*m) return;
    }
    fCol ++;
    // Bottom wall from left to right
```

```
for (int j = fCol; j <= lCol; j++){
    System.out.println(mat[lRow][j]);
    count++;
    if(count == n*m) return;
}
lRow--;
// Right wall from bottom to top
for(int i = lRow; i >= fRow; i--){
    System.out.println(mat[i][lCol]);
    count++;
    if(count == n*m) return;
}
lCol--;
for(int j = lCol; j >= fCol; j--){
    System.out.println(mat[fRow][j]);
    count++;
    if(count==n*m) return;
}
fRow++;
}
```

(4) Exit Point: on 0, move same way, on 1, move 90° right.

Constructive algorithm.





$$dx = \{1, 0, -1, 0\}$$
$$dy = \{0, 1, 0, -1\}$$