Print Alternate Row

rows = 4

cols = 6

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

o/p

2 3 8 7 0 4

0 0 8 1 0 8

Code

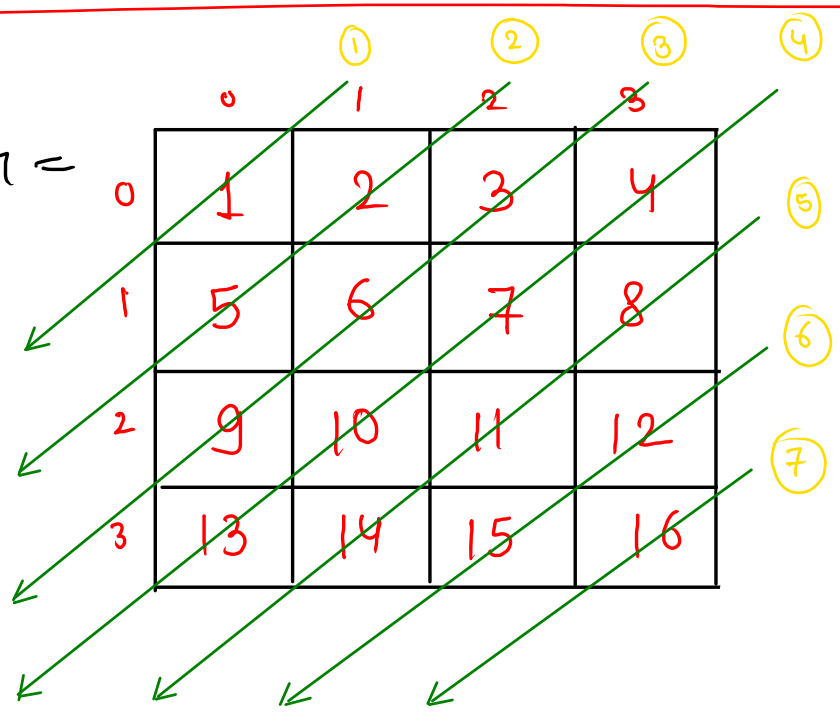
```
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int rows = scn.nextInt();
    int cols = scn.nextInt();
    int[][] arr = new int[rows][cols];
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            arr[i][j] = scn.nextInt();
        }
    }
    printMatrix(arr, rows, cols);
}
public static void printMatrix(int[][] arr, int rows, int cols) {
    for (int i = 0; i < rows; i += 2) {
        for (int j = 0; j < cols; j++) {
            System.out.print( arr[i][j] + " " );
        }
        System.out.println();
    }
}
```

4 6

Print the matrix left-diagonal wise

(it is a square matrix only)

arr =



output :- 1 2 5 3 6 9 4 7 10 13 8 11 14 12 15 16

logic

gap

arr =

	0	1	2	3
0	1	2	3	4
1	5	6	7	8
2	9	10	11	12
3	13	14	15	16

for (initialisation; condⁿ; upgrade) {

}

first half

upgradation :- $i{+}{+}$, $j{-}{-}$

Condition :- $j >= 0$

initialization :- $(0,0)$, $(0,1)$, $(0,2)$, $(0,3)$

(i, j)

$i = 0$, $j = 0, 1, 2, 3$

gap

second half

$\boxed{n = 4}$

upgradation :- $i{+}{+}, j{-}{-}$

cond^n :- $i < n$

initialization :- $(1, 3), (2, 3), (3, 3)$

$j = n-1, \quad i = 1, 2, 3$

gap

Code

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int[][] arr = new int[n][n];
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            arr[i][j] = scn.nextInt();
        }
    }
    printLeftDiagonal(arr, n);
}
public static void printLeftDiagonal(int[][] arr, int n) {
    for (int gap = 0; gap < n; gap++) {
        for (int i = 0, j = gap; j >= 0; i++, j--) {
            System.out.print(arr[i][j] + " ");
        }
    }

    for (int gap = 1; gap < n; gap++) {
        for (int i = gap, j = n - 1; i < n; i++, j--) {
            System.out.print(arr[i][j] + " ");
        }
    }
}
```

first half

second half

oper = $n*n$

$$\boxed{\begin{aligned} T.C &= O(n*n) \\ &= O(n^2) \end{aligned}}$$

code

(n = 3)

```java
public static void printLeftDiagonal(int[][] arr, int n) {
    for (int gap = 0; gap < n; gap++) {
        for (int i = 0, j = gap; j >= 0; i++, j--) {
            System.out.print(arr[i][j] + " ");
        }
    }

    for (int gap = 1; gap < n; gap++) {
        for (int i = gap, j = n - 1; i < n; i++, j--) {
            System.out.print(arr[i][j] + " ");
        }
    }
}
```

arr

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

op:- 1 2 4 3 5 7 6 8 9

(i,j)

gap = 0, (0,0)
(1,-1) ×

gap = 1, (0,1)
(1,0)
(2,-1) ×

gap = 2, (0,2)
(1,1)
(2,0)
(3,-1) ×

(i,j)

gap = 1, (1,2)
(2,1)
(3,0) ×

gap = 2, (2,2)
(3,1) ×

H.W.

try yourself

arr =

	0	1	2	3
0	1	2	3	4
1	5	6	7	8
2	9	10	11	12
3	13	14	15	16

o/p:- 4 3 8 2 7 12 1 6 11 16 5 10 15 9 14 13

⇒ Transpose (convert all rows into cols and vice versa)

arr =

	0	1	2	3
0	1	2	3	4
1	5	6	7	8
2	9	10	11	12
3	13	14	15	16

→

	0	1	2	3
0	1	5	9	13
1	2	6	10	14
2	3	7	11	15
3	4	8	12	16

$S.C = O(1)$