

1. Interchange the rows.

3	-n
3	-m
8	1 0
9	9 6
6	6 4

⇒

6	6	4
9	9	6
8	1	0

2 ways:

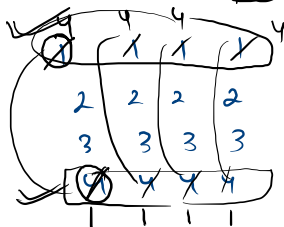
1D.

↗

last (n-1)



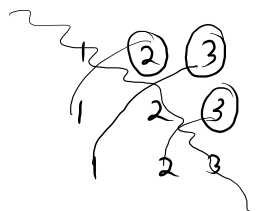
eg. \int
tmp = 1
✓



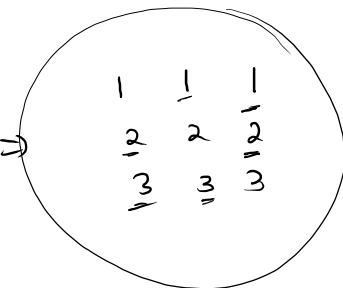
⇒

4	4	4	4
2	2	2	2
3	3	3	3
1	1	1	1

2. Transpose 2D mat ($n \times n$)



⇒

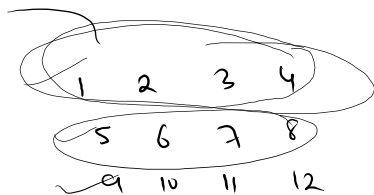


half of sq.



1D

3.

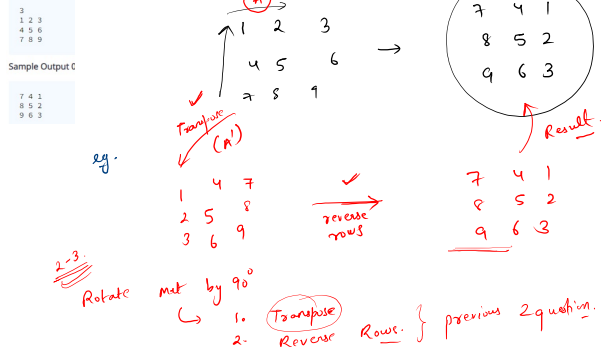


⇒

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

Rotate The Matrix by 90 Degree

Take a square matrix of size $n \times n$ as input, and rotate the matrix by 90 degree.



```

import java.io.*;
import java.util.*;

public class Solution {
    public static void transpose(int[][] A){
        int n = A.length;
        for(int i = 0; i < n; i++){
            for(int j = 1; j < n; j++){
                int tmp = A[i][j];
                A[i][j] = A[j][i];
                A[j][i] = tmp;
            }
        }
    }

    public static void reverseRows(int[][] A){
        int n = A.length;
        for(int row = 0; row < n; row++){
            int i = 0;
            int j = n-1;
            while(i < j){
                int tmp = A[row][i];
                A[row][i] = A[row][j];
                A[row][j] = tmp;
                i++;
                j--;
            }
        }
    }

    public static void printArr(int[][] A){
        int n = A.length;
        for(int i = 0; i < n; i++){
            for(int j = 0; j < n; j++){
                System.out.print(A[i][j] + " ");
            }
            System.out.println();
        }
    }

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();

        int[][] A = new int[n][n];
        for(int i = 0; i < n; i++){
            for(int j = 0; j < n; j++){
                A[i][j] = scn.nextInt();
            }
        }

        //1. transpose
        transpose(A);
        //2. reverse rows
        reverseRows(A);
        //3. display
        printArr(A);
    }
}

```

Convert 1-D Array to 2-D Array

Take an array of size N as input, representing a 1-D array.

There are many possible factors of N , for eg:- $p \times q = N$.

Now take p and q as input and print the 2-D array with dimensions as $p \times q$.

Note: It is guaranteed that a 2-D array will be formed.

Sample Input 0

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

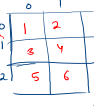
Sample Output 0

1 2 3
4 5 6
7 8 9

Sample Input 1

6 5 4 3 2 1
Sample Output 1

1 2
3 4
5 6



3x2

?



3x3

PxQ

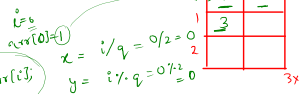
Logic

eg. $n=6$
arr



$x = i/q$
 $y = i \% q$
 $A[x][y] = arr[i]$

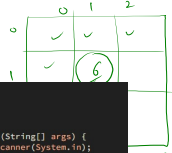
$p=3$
 $q=2$



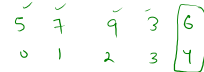
2. For 2D arr.

$x = i/q$
 $y = i \% q$
 $A[x][y] = arr[i]$

$i=2$
 $x = 2/2 = 1$
 $y = 2 \% 2 = 0$



P Q
3 2



$i=4/3 = 1$
 $i=4 \% 3 = 1$

```
public class Solution {
    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        int[] arr = new int[n];
        for(int i = 0; i < n; i++){
            arr[i] = scn.nextInt();
        }
        int p = scn.nextInt();
        int q = scn.nextInt();

        //Logic
        //create 2D array of size p * q
        int[][] A = new int[p][q];
        //fill
        for(int i = 0; i < n; i++){
            int x = i / q;
            int y = i % q;
            A[x][y] = arr[i];
        }
        //print A
        for(int i = 0; i < p; i++){
            for(int j = 0; j < q; j++){
                System.out.print(A[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

```
import java.util.*;
import java.util.*;

public class Solution {
    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        int[] arr = new int[n];
        for(int i = 0; i < n; i++){
            arr[i] = scn.nextInt();
        }
        int p = scn.nextInt();
        int q = scn.nextInt();

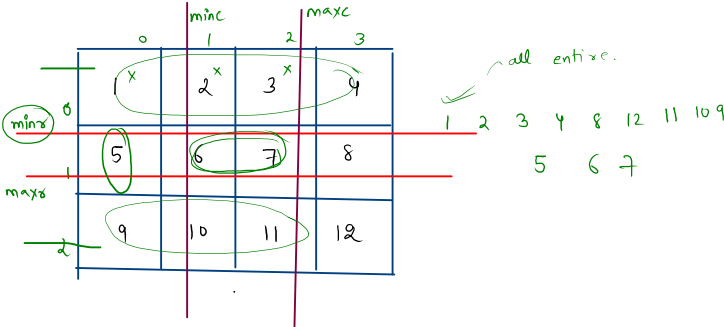
        //Logic
        //create 2D array of size p * q
        int[][] A = new int[p][q];
        //fill
        for(int i = 0; i < n; i++){
            int x = i / q;
            int y = i % q;
            A[x][y] = arr[i];
        }
        //print A
        for(int i = 0; i < p; i++){
            for(int j = 0; j < q; j++){
                System.out.print(A[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

Spiral Matrix 44

Print all the elements of a m*n matrix in the spiral form as shown below. Note: Start traversing from the - (0th row and 0th column)

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

Input: matrix = [[1,2,3,4],[5,6,7,8],[9,10,11,12]]
Output: [1,2,3,4,8,12,11,10,9,5,6,7]



```
for(int j = minc; j <= maxc; j++){
    System.out.print(A[0][j] + " ");
}
//left
for(int i = minx; i <= maxx; i++){
    System.out.print(A[i][maxc] + " ");
}
//down
for(int j = maxc; j >= minc; j--){
    System.out.print(A[maxx][j] + " ");
}
//up
for(int i = maxx; i >= minx; i--){
    System.out.print(A[i][minc] + " ");
}
minc++;
maxc--;
minx++;
maxx--;
```

The diagram shows a 3x4 matrix with spiral traversal paths. The matrix is divided into four quadrants by a horizontal line (minx=1) and a vertical line (maxc=2). The spiral path is indicated by green arrows: 1 (top-left), 2 (top), 3 (top-right), 4 (right), 8 (bottom-right), 12 (bottom), 11 (bottom-left), 10 (left), 9 (top-left), 5 (left), 6 (top), 7 (top-right). The indices minc, maxc, minx, and maxx are marked on the axes. The output sequence is shown as 1 2 3 4 8 12 11 10 9 5 6 7.

```
for(int j = minc; j <= maxc; j++){
    System.out.print(A[0][j] + " ");
}
//left
for(int i = minx; i <= maxx; i++){
    System.out.print(A[i][maxc] + " ");
}
//down
for(int j = maxc; j >= minc; j--){
    System.out.print(A[maxx][j] + " ");
}
//up
for(int i = maxx; i >= minx; i--){
    System.out.print(A[i][minc] + " ");
}
minc++;
maxc--;
minx++;
maxx--;
```

The diagram shows a 3x4 matrix with spiral traversal paths. The matrix is divided into four quadrants by a horizontal line (minx=1) and a vertical line (maxc=2). The spiral path is indicated by green arrows: 1 (top-left), 2 (top), 3 (top-right), 4 (right), 8 (bottom-right), 12 (bottom), 11 (bottom-left), 10 (left), 9 (top-left), 5 (left), 6 (top), 7 (top-right). The indices minc, maxc, minx, and maxx are marked on the axes. The output sequence is shown as 1 2 3 4 8 12 11 10 9 5 6 7.

```

import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int row = scn.nextInt();
        int col = scn.nextInt();

        int [][] A = new int[row][col];
        for(int i = 0; i < row; i++){
            for(int j = 0; j < col; j++){
                A[i][j] = scn.nextInt();
            }
        }
        // define 4 walls
        int minr = 0;
        int maxr = row - 1;

        int minc = 0;
        int maxc = col - 1;

        int total = row * col;
        int count = 0;    // how many printed

        while(count < total){
            for(int j = minc; j <= maxc && count < total; j++){
                System.out.print(A[minr][j]+" ");
                count++;
            }
            minr++;
            for(int i = minr; i <= maxr && count < total; i++){
                System.out.print(A[i][maxc]+" ");
                count++;
            }
            maxc--;
            for(int j = maxc; j >= minc && count < total; j--){
                System.out.print(A[maxr][j]+" ");
                count++;
            }
            maxr--;
            for(int i = maxr; i >= minr && count < total; i--){
                System.out.print(A[i][minc]+" ");
                count++;
            }
            minc++;
        }
    }
}

```