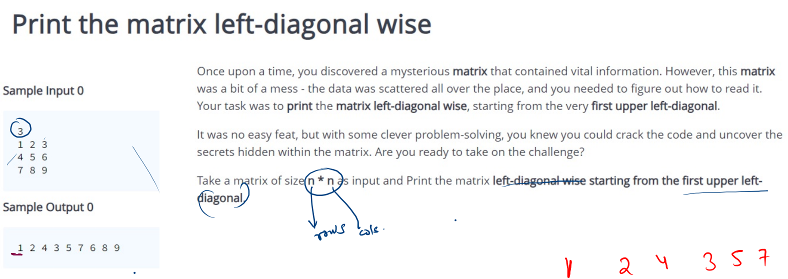


Print the matrix left-diagonal wise

Once upon a time, you discovered a mysterious **matrix** that contained vital information. However, this **matrix** was a bit of a mess - the data was scattered all over the place, and you needed to figure out how to read it. Your task was to **print** the **matrix left-diagonal wise**, starting from the very **first upper left-diagonal**.

It was no easy feat, but with some clever problem-solving, you knew you could crack the code and uncover the secrets hidden within the matrix. Are you ready to take on the challenge?

Take a matrix of size n * n as input and Print the matrix **~~left-diagonal wise~~ starting from the first upper left-diagonal**

rows cols.

Sample Input 0

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

Sample Output 0

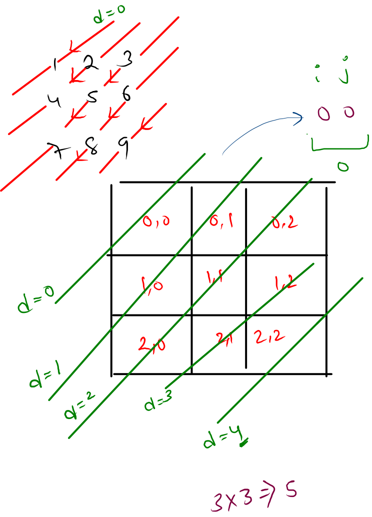
```
1 2 4 3 5 7 6 8 9
```

1 2 4 3 5 7 6 8 9

i j

0 0 | 0,1 1,0 | 0,2. 1,1 2,0 | 1,2 2,1 | 2,2

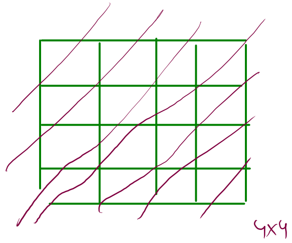
0 | 1 | 2 | 3 | 4



3x3 ⇒ 5

n*n.

$\boxed{2n-1}$



4x4 = 7

3x3

↳ d=5 0 1 2 3 4 < d

4x4

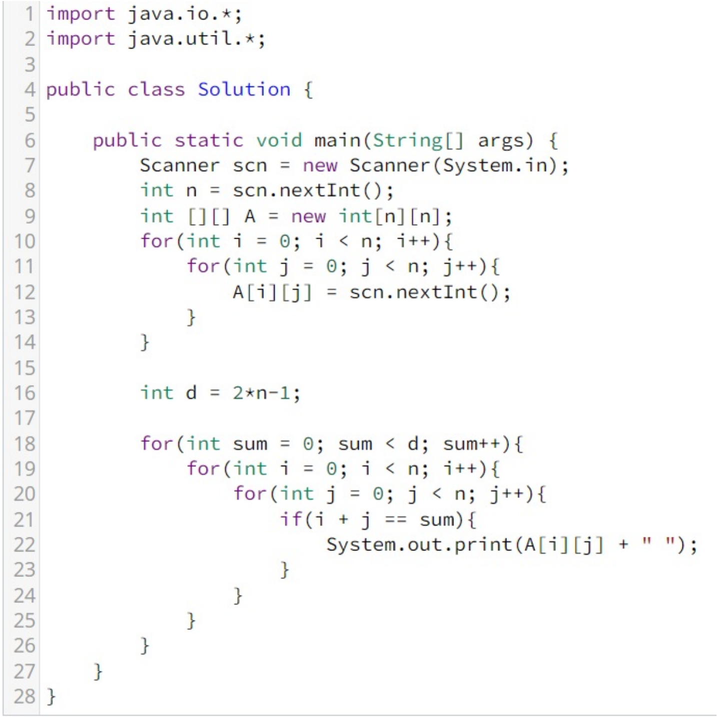
d=7 0 1 2 3 4 5 6 < d.

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

n=3

d = 2n-1 = 5

sum = ∅ 1 < 5 ✓



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

public class Solution {

    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();
            }
        }

        int d = 2*n-1;

        for(int sum = 0; sum < d; sum++){
            for(int i = 0; i < n; i++){
                for(int j = 0; j < n; j++){
                    if(i + j == sum){
                        System.out.print(A[i][j] + " ");
                    }
                }
            }
        }
    }
}
```

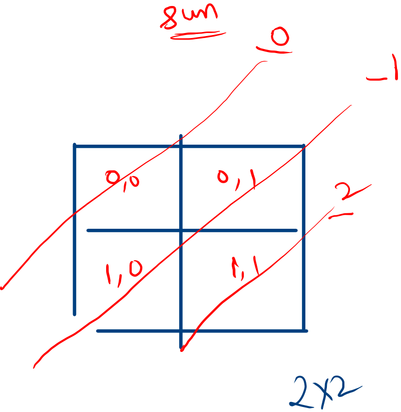
i = ∅

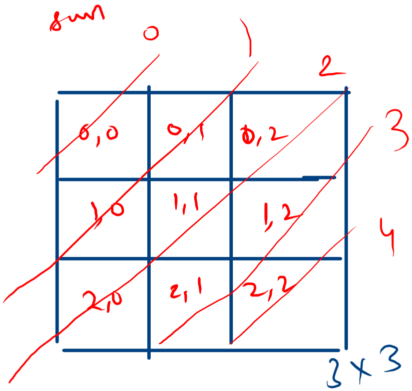
i = 1

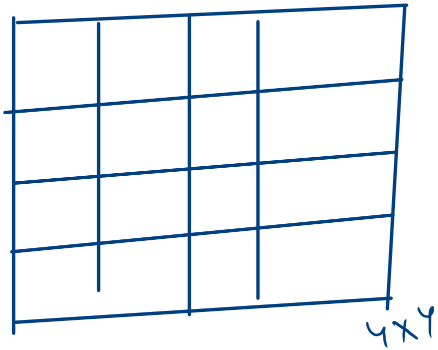
j = 0

1+0 == 1 ✓

1 2 ✓ 4 ....







3

5

7

sum < d.

0 1 2 3 4 < 5.

sum < d.

Print Upper triangular matrix 1

In the world of finance, every second counts. That's why a young financial analyst named Maria is given a complex **matrix** of size **m * n** to analyze, she knows that time is of the essence. The **matrix** contains vital data that could make or break her company's fortunes, and Maria has to act fast to make sense of it all.

As she delves into the **matrix**, Maria realizes that the **upper triangle** holds the key to unlocking the data's secrets.

help Maria and create program that **print** the **upper triangular matrix**.

Sample Input 0

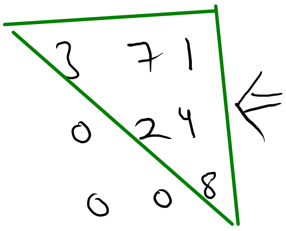
```
3
3
3 7 1
6 2 4
7 1 0
```

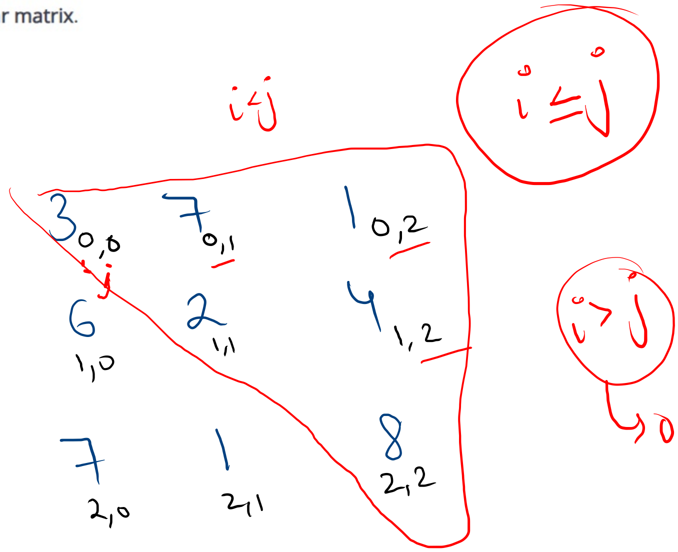
3

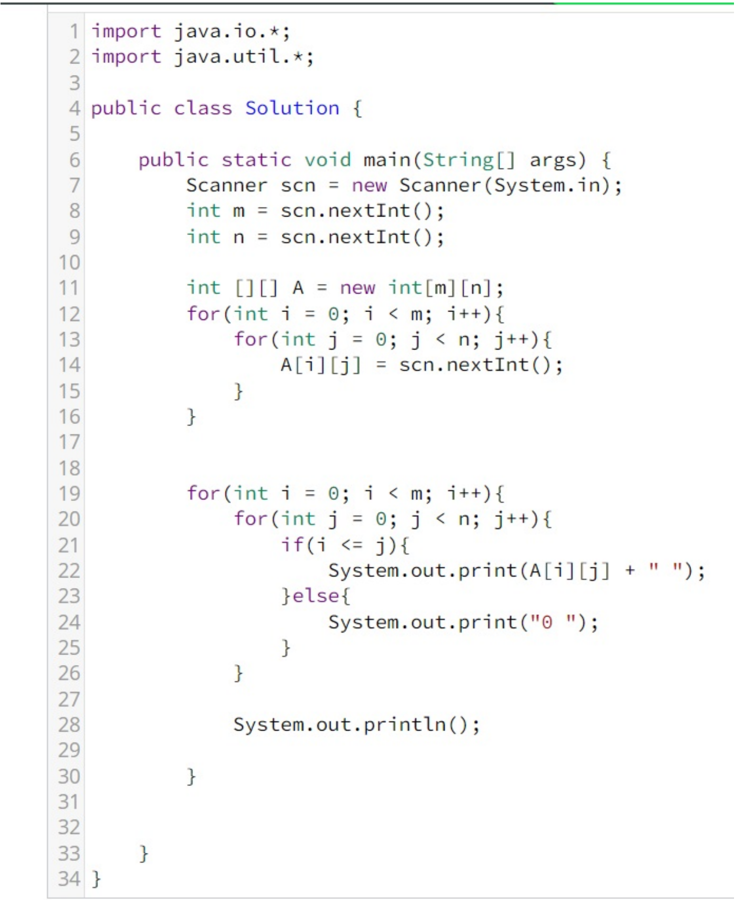
3

Sample Output 0

```
3 7 1
0 2 4
0 0 0
```



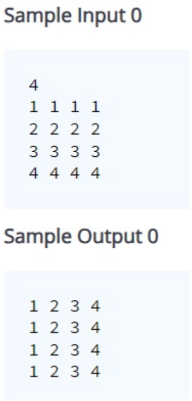




```java
import java.io.*;
import java.util.*;

public class Solution {

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

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


        for(int i = 0; i < m; i++){
            for(int j = 0; j < n; j++){
                if(i <= j){
                    System.out.print(A[i][j] + " ");
                }else{
                    System.out.print("0 ");
                }
            }

            System.out.println();

        }


    }
}
```

Transpose of Matrix of N*N

$n = 4$



Sample Input 0

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

Sample Output 0

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

$$\begin{matrix} 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 \end{matrix} \xrightarrow{\text{Transpose}} \begin{matrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \end{matrix}$$

→ interchanging rows & columns.

22 ~~22~~ 0,0	32 ~~23~~ 0,1	42 ~~24~~ 0,2	~~25~~ 52 0,3
23 ~~32~~ 1,0	33 ~~33~~ 1,1	43 ~~34~~ 1,2	~~35~~ 53 1,3
24 ~~42~~ 2,0	34 ~~43~~ 2,1	44 ~~44~~ 2,2	54 ~~45~~ 2,3
25 ~~52~~ 3,0	35 ~~53~~ 3,1	45 ~~54~~ 3,2	~~55~~ 55 3,3

2≤3

2≤2

2≤1

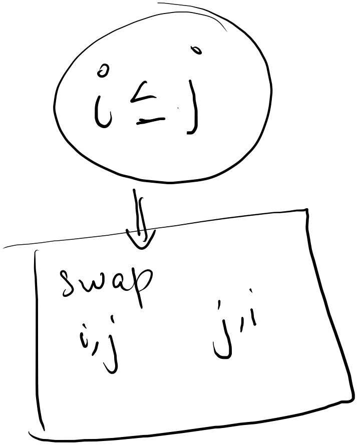
2≤0

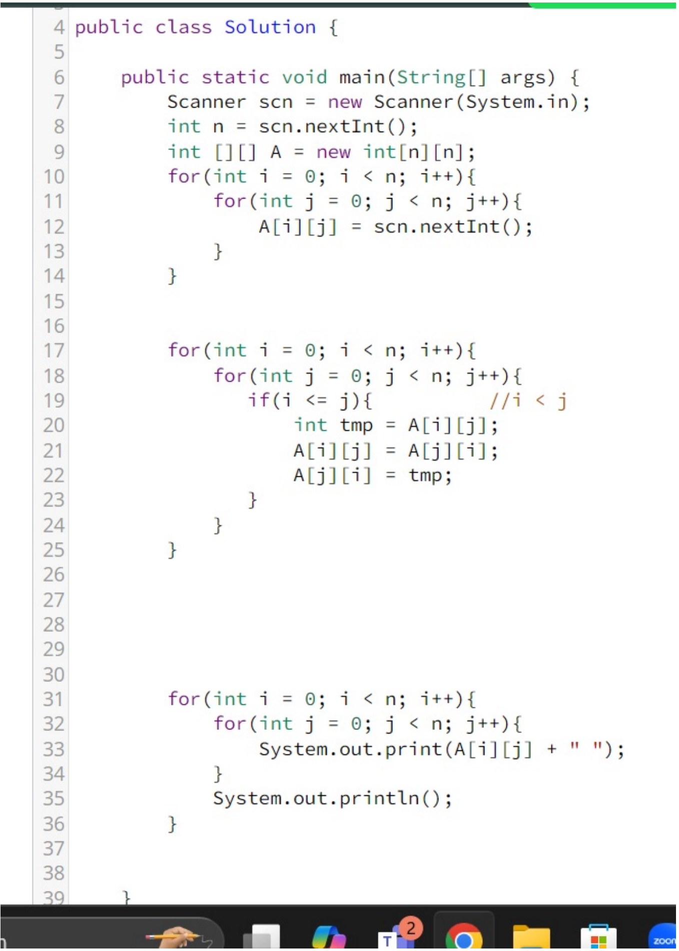
1≤2

1≤3

ans

22 0,0	32 0,1	42 0,2	52 0 3
23 1 0	33 1 1	43 1 2	53 1 3
~~3~~24 2 0	34 2 1	44 2 2	54 2 3
25 3 0	35 3 1	45 3 2	55 3,3



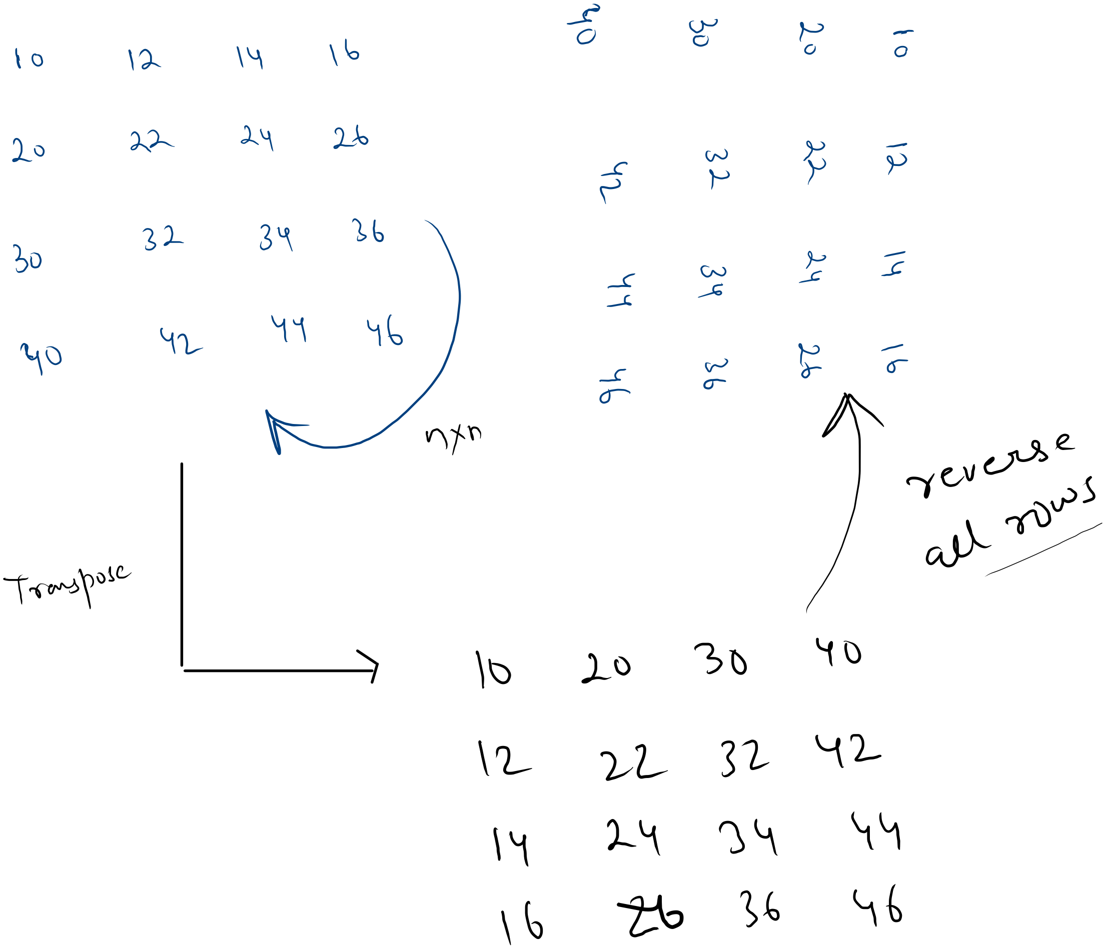


```
public class Solution {

    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();
            }
        }


        for(int i = 0; i < n; i++){
            for(int j = 0; j < n; j++){
               if(i <= j){          //i < j
                   int tmp = A[i][j];
                   A[i][j] = A[j][i];
                   A[j][i] = tmp;
               }
            }
        }





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


    }
```

Rotate The Matrix by 90 Degree

Rotate by 90°

1. Transpose
2. Reverse all rows



10	12	14	16
20	22	24	26
30	32	34	36
40	42	44	46

10	20	30	40
12	22	32	42
14	24	34	44
16	26	36	46

40	30	20	10
42	32	22	12
44	34	24	14
46	36	26	16

row →

4 0

	40	30	20	10
row	~~10~~ 0,$\boxed{0}$	~~20~~ 0,1	~~30~~ 0,2	~~40~~ 0$\boxed{3}$

j i

row = 0

1

2

3 < n

[0][i] ↔ [0][j]

11 1,0	12 1,1	13 1,2	14 1,3
13 2,0	16 2,1	19 2,2	18 2,3
13 3,0	16 3,1	19 3,2	18 3,3

row = 0

tmp=81

83 ~~80~~ 82 ~~81~~ 81 ~~82~~ ~~83~~ 80

0,0 0,1 0,2 0,3

1<2 ✓

~~i~~ ~~j~~

j i

tmp = A[r][i] A[0][1]

A[r][i]

row

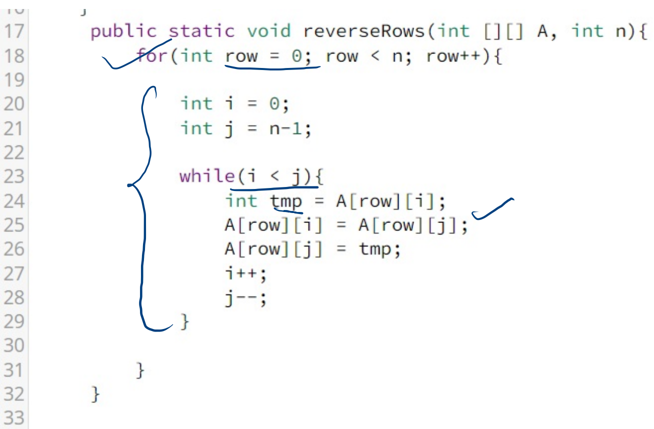
A[0][1] = A[0][2]

A[0][2] =

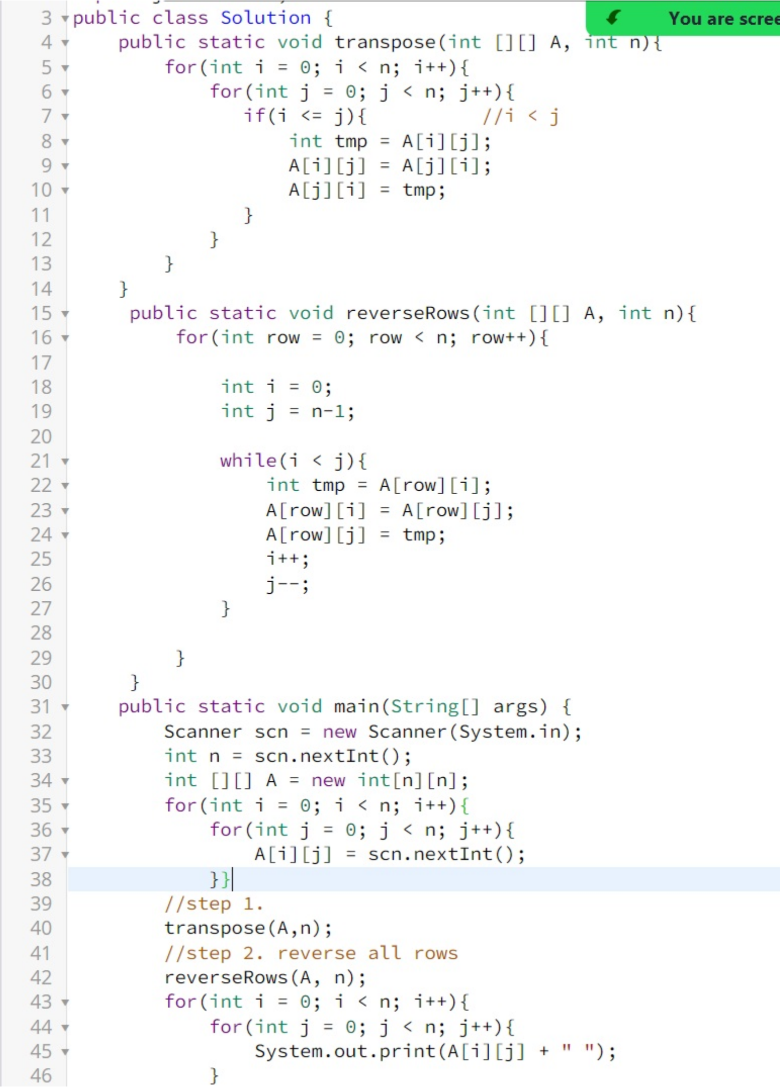
70 71 72 73

60 61 62 62

20 21 22 23



```
public static void reverseRows(int [][] A, int n){
    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 class Solution {
    public static void transpose(int [][] A, int n){
        for(int i = 0; i < n; i++){
            for(int j = 0; j < n; j++){
               if(i <= j){          //i < j
                   int tmp = A[i][j];
                   A[i][j] = A[j][i];
                   A[j][i] = tmp;
               }
            }
        }
    }
     public static void reverseRows(int [][] A, int n){
         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 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();
            }}
        //step 1.
        transpose(A,n);
        //step 2. reverse all rows
        reverseRows(A, n);
        for(int i = 0; i < n; i++){
            for(int j = 0; j < n; j++){
                System.out.print(A[i][j] + " ");
            }
```

5 4 3 2 1

```
for ( i=n-1    i≥0    i--)
        print (A(i) .
```

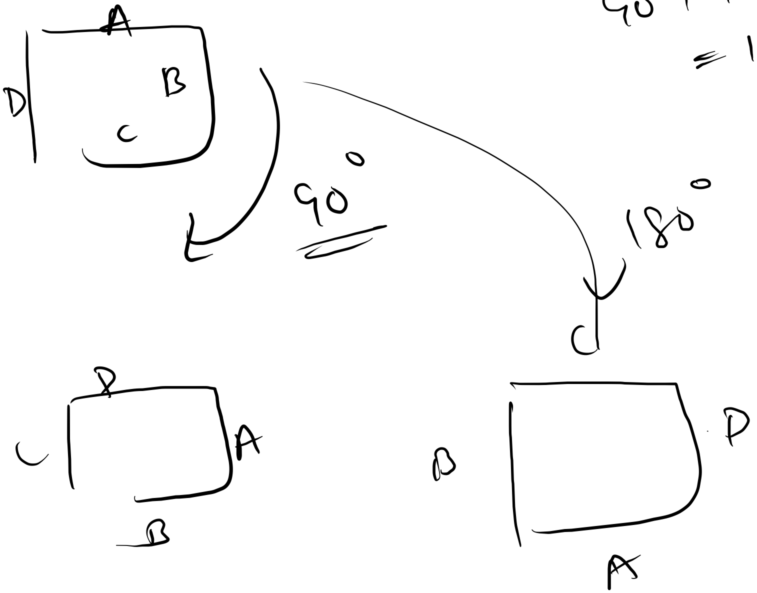
$A[i] = 2$

1	2	3	4	5
0	1	2	3	4

$A[i]=2$.

rotate90°()

90 + 90
= 180


A
D
B
C
90°
180°
D
C
A
B
C
D
B
A