Rotate 180°

$$\begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{matrix} \xrightarrow{90^\circ} \begin{matrix} \text{[illegible]} & \text{[illegible]} & \text{[illegible]} \\ \text{[illegible]} & \text{[illegible]} & \text{[illegible]} \\ \text{[illegible]} & \text{[illegible]} & \text{[illegible]} \end{matrix} \xrightarrow{180^\circ} \begin{matrix} \text{[illegible]} & \text{[illegible]} & \text{[illegible]} \\ \text{[illegible]} & \text{[illegible]} & \text{[illegible]} \\ \text{[illegible]} & \text{[illegible]} & \text{[illegible]} \end{matrix}$$

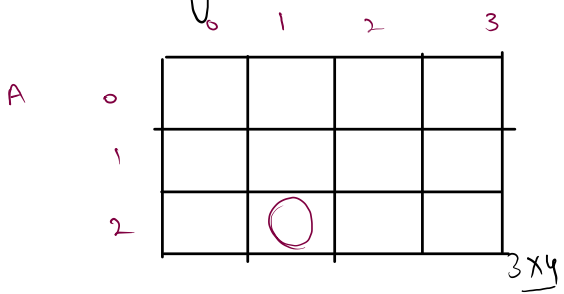
$$A \xrightarrow{90^\circ} A' \xrightarrow{90^\circ} A''$$

$$\boxed{A'' \text{ is } 180^\circ \text{ of } A}$$

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

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 rotateBy90(int [][] A, int n){
        transpose(A,n);
        reverseRows(A, n);
    }
```

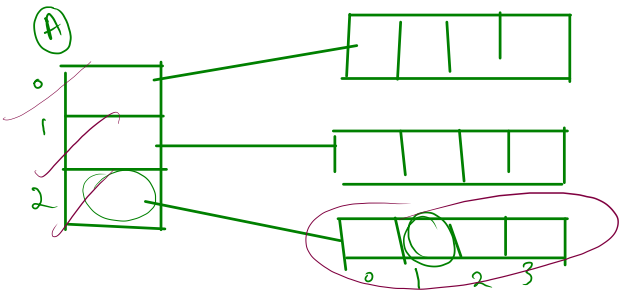
```java
    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        int [][] A = new int[n][n];
        for(int i = 0; i < n; i++){
            for(int j = 0; j < n; j++){
                A[i][j] = scn.nextInt();
            }
        }


        //step 1.
        //step 2. reverse all rows
        rotateBy90(A, n);
        rotateBy90(A, n);

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


    }
}
```

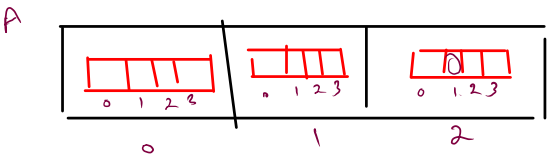
2D- Array.



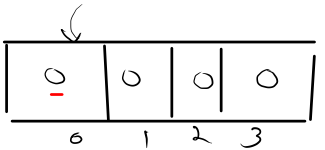
A[2][1]

A[2][1]





int [ ] A = new int [4];



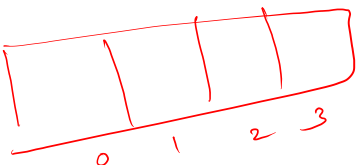
summary

A.length = 3 = rows.

A[0].length = 4 = cols.

A[2].length

A[-1].length.



2022. Convert 1D Array Into 2D Array

Easy 783 56 Add to List Share

You are given a **0-indexed** 1-dimensional (1D) integer array `original`, and two integers, `m` and `n`. You are tasked with creating a 2-dimensional (2D) array with `m` rows and `n` columns using **all** the elements from `original`.

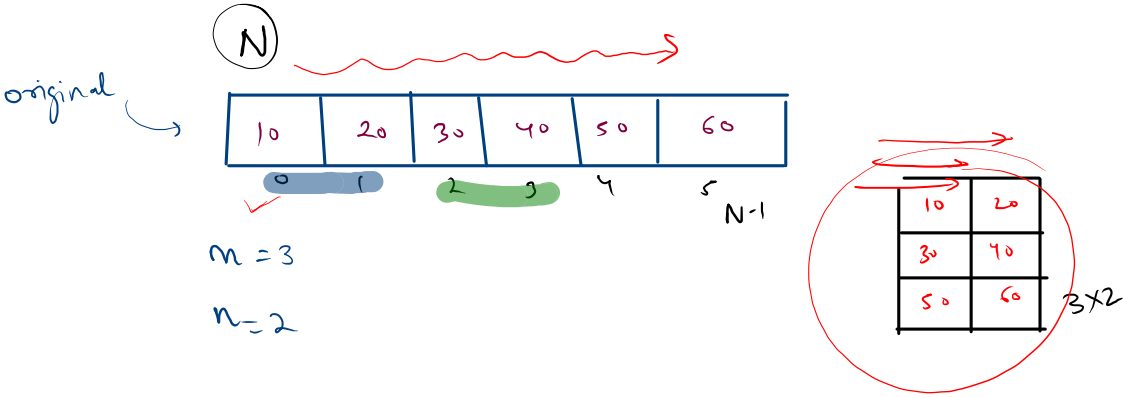
The elements from indices `0` to `n - 1` (**inclusive**) of `original` should form the first row of the constructed 2D array, the elements from indices `n` to `2 * n - 1` (**inclusive**) should form the second row of the constructed 2D array, and so on.

Return *an* `m x n` *2D array constructed according to the above procedure, or an empty 2D array if it is impossible.*

```
class Solution {
    public int[][] construct2DArray(int[] original, int m, int n) {

    }
}
```

10 20

3



n = 2

2n-1

```java
class Solution {
    public int[][] construct2DArray(int[] original, int m, int n) {
        if(m*n != original.length){
            return new int[][]{};
        }

        int[][] A = new int[m][n];
        int k = 0;

        for(int i = 0; i < m; i++){
            for(int j = 0; j < n; j++){
                A[i][j] = original[k];
                k++;
            }
        }


        return A;
    }
}
```

original.

10	20	30	40	50	60
0	1	2	3	4	5
		k			

m = 3

n = 2

i → 0 < 3

j < 2

j = 2

2 < 2

	0	1
0	10	20
1		
2		

3 x 2

```java
class Solution {
    public int[][] construct2DArray(int[] original, int m, int n) {
        if(m*n != original.length){
            return new int[][]{};
        }

        int[][] A = new int[m][n];
        int k = 0;

        for(int i = 0; i < m; i++){
            for(int j = 0; j < n; j++){
                A[i][j] = original[k];
                k++;
            }
        }


        return A;
    }
}
```

Print row wise with condition

Once upon a time, there was a programmer named Alex who was given the task of printing a **matrix row-wise**. However, there was a twist - the **even-numbered rows** had to be printed from **left to right**, and the **odd-numbered rows** had to be printed from **right to left.**

help Alex and write a program that would **iterate** through each **row** of the **matrix** and check if it was an **even or odd row.** If it was an **even row**, the program would traverse it from **left to right**, and if it was an **odd row**, the program would traverse it from **right to left.**

3
4

e 0	11	12	13	14
o 1	21	22	23	24
e 2	31	32	33	34

11 12 13 14

24 23 22 21

31 32 34 34

```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++){

            if(i % 2 == 0){
                for(int j = 0; j < n; j++){
                    System.out.print(A[i][j] + " ");
                }
            }else {
                for(int j = n-1; j >= 0; j--){
                    System.out.print(A[i][j] + " ");
                }
            }

            System.out.println();

        }
```

Shift Matrix Row-Wise

Once upon a time, there was a group of students who were working on a project to design a gaming platform. They had a **2D grid** of **N * N** size which represented the game board. Each cell of the grid had some data associated with it.

One day, they encountered a problem where they had to shift the elements of the grid **row-wise** in clock wise direction by a certain number of positions, **k**. This was necessary to create a more interesting and challenging gaming experience for their users.

The students decided to write a Java program to solve this problem. They came up with an algorithm to **shift** the elements of the **grid row-wise by k positions**. After implementing the algorithm, they were able to shift the elements of each **row** by **k** positions.

Write a program that shift each row of matrix by **k**.

Sample Input 0

```
3
0 5 9
2 7 5
2 3 3
2
```

0 5 9
2 7 5
2 3 8

0 5 9 — 9 0 5
5 2 7
8 2 3

Sample Output 0

```
9 0 5
5 2 7
3 2 3
```

k=2

n=5

11	12	13	14	15
21	22	23	24	25
16	17	18	19	20
41	42	43	44	45
61	62	63	64	65

13	14	15	11	12
23	24	25	21	22
18	19	20	16	17
43	44	45	41	42
63	64	65	61	62