Revision

$$\begin{tikzcd}
\begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{matrix} \arrow[loop above, "90^\circ"] \arrow[rr, bend left, "\text{Transpose}"] \arrow[dr, bend right, "90^\circ"'] & & \begin{matrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{matrix} \arrow[dl, bend left, "\text{reverse rows}"] \\
& \boxed{\begin{matrix} 7 & 4 & 1 \\ 8 & 5 & 2 \\ 9 & 6 & 3 \end{matrix}} &
\end{tikzcd}$$

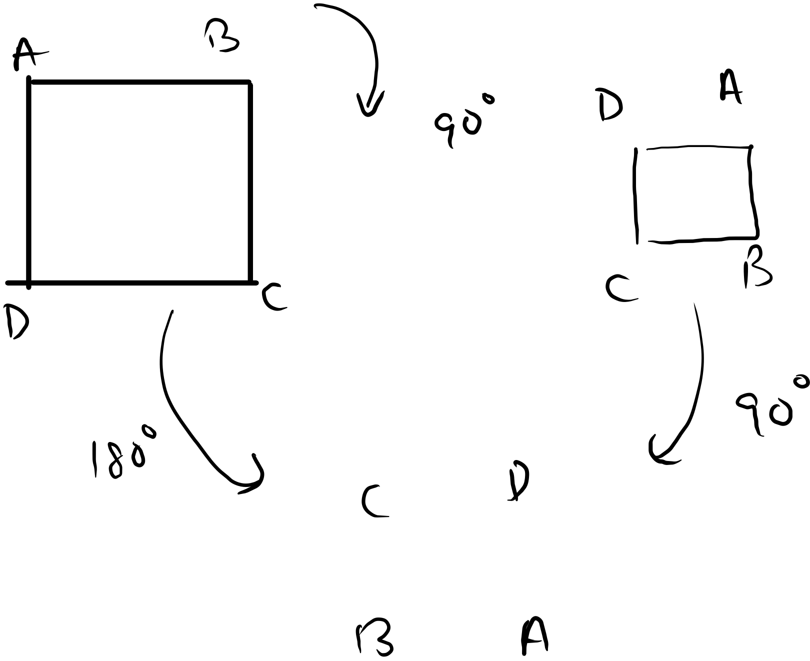
ans.

180° ... ?

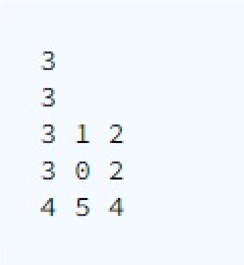
270° ... ?

180° ... 90° .. 2 times.

270° ... 90° ... 3 times.

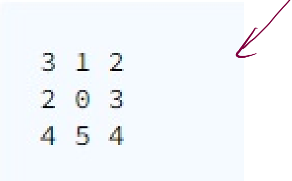


Print row wise with condition



```
3
3
3 1 2
3 0 2
4 5 4
```

Sample Output (



```
3 1 2
2 0 3
4 5 4
```

3 1 2
2 0 3
4 5 4

rows

	0	1	2
0	3	1	2
1	3	0	2
2	4	5	4

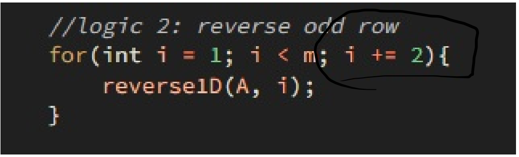
→ reverse row.

row 0 → c-1
LR

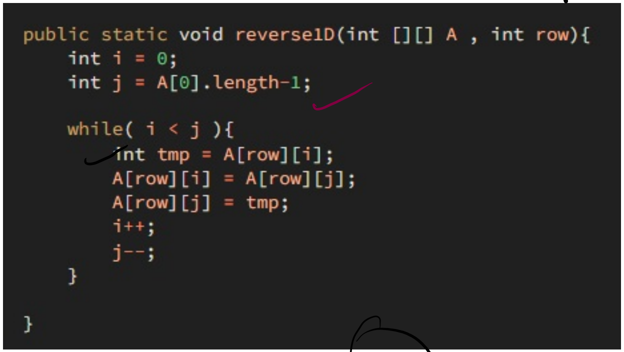
1 ← c-1
RL

row → even → LR
└→ odd → RL

logic 2.



```
//logic 2: reverse odd row
for(int i = 1; i < m; i += 2){
    reverse1D(A, i);
}
```



```
public static void reverse1D(int [][] A , int row){
    int i = 0;
    int j = A[0].length-1;

    while( i < j ){
        int tmp = A[row][i];
        A[row][i] = A[row][j];
        A[row][j] = tmp;
        i++;
        j--;
    }

}
```

i=1

x i<j

c = A[r][i]

A[1][0]

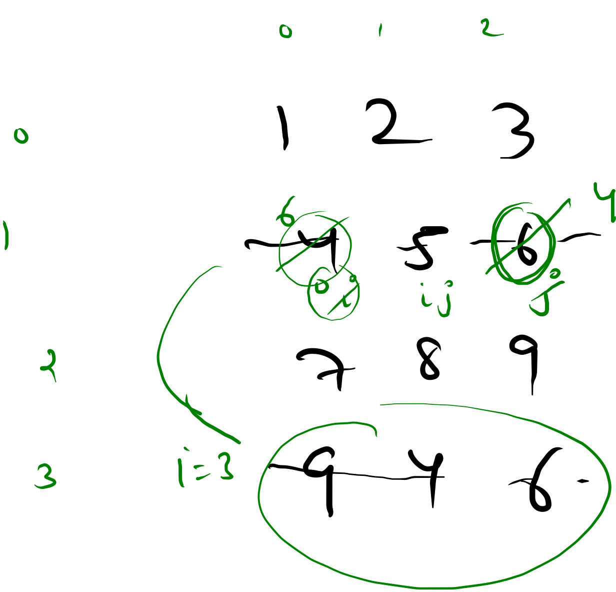
t = A[1] [0] = 4

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

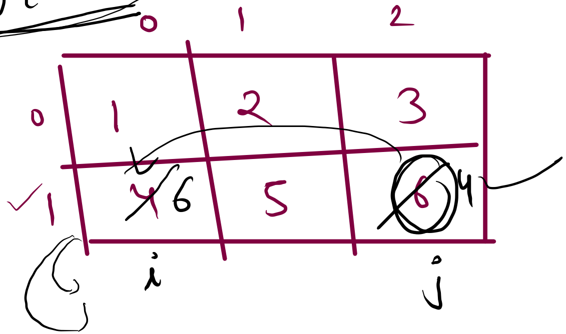
A[1][2]

A(r)[j] = tmp

rw →



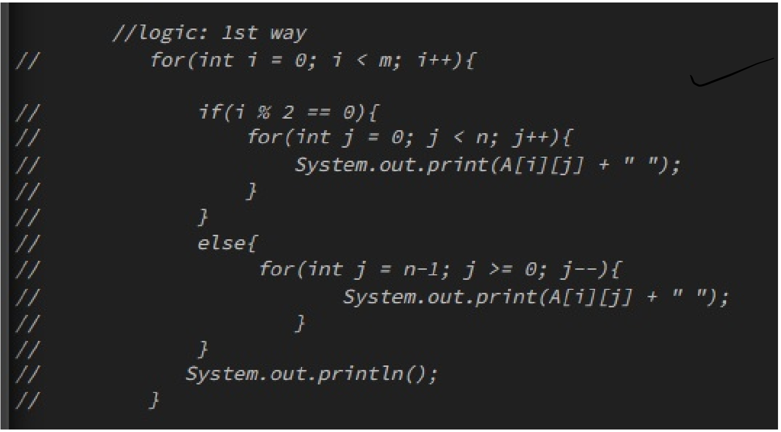
0 < 2 ✓



i = 0

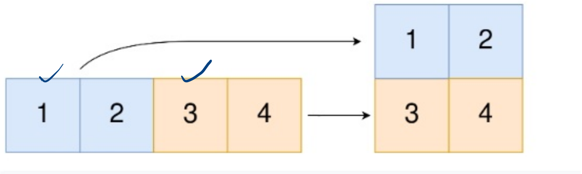
j = 2

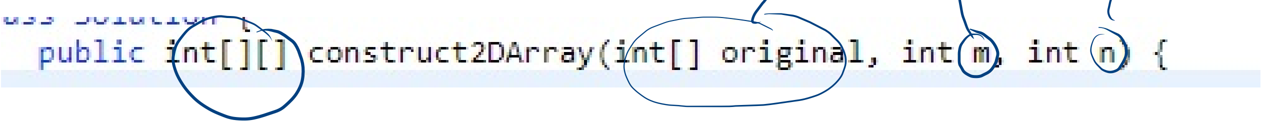
logic - 1



```
        //logic: 1st way
//         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();
//         }
```

2022. Convert 1D Array Into 2D Array





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

1D

row

n

return

2 2

1	2
3	

2x2

row - major

eg. n=5

1	2	3	4	5

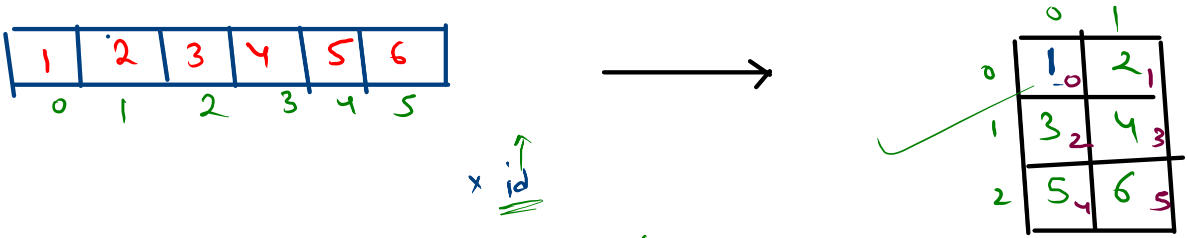
5 cells.

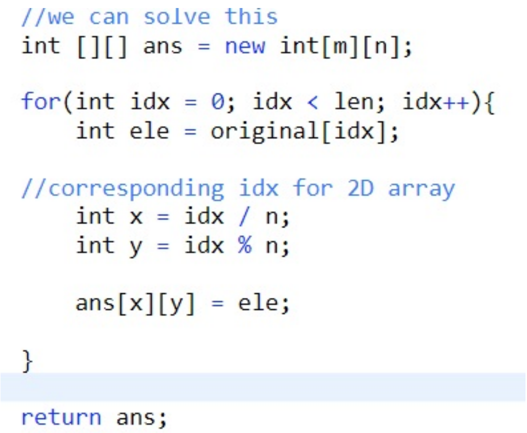
2D

m=2 n=2

1	2
3	4

4 cells





```
//we can solve this
int [][] ans = new int[m][n];

for(int idx = 0; idx < len; idx++){
    int ele = original[idx];

//corresponding idx for 2D array
    int x = idx / n;
    int y = idx % n;

    ans[x][y] = ele;

}

return ans;
```

ele = 6

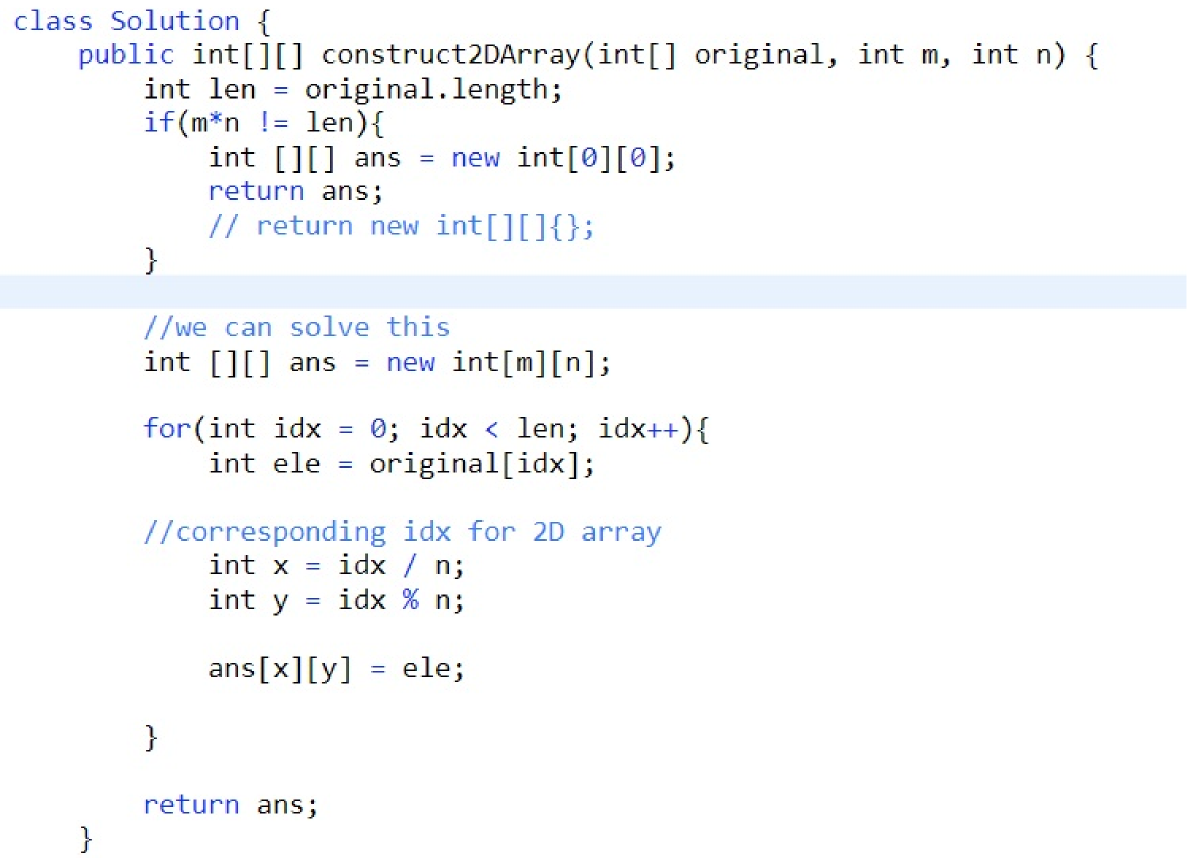
$x = 5/2 = 2$

$y = 8\%2 = \phi$

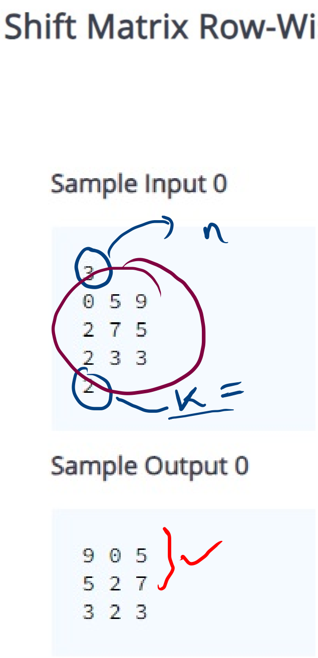
0, 0

$x = idx / n$ — cols

$y = idx \% n$



```
class Solution {
    public int[][] construct2DArray(int[] original, int m, int n) {
        int len = original.length;
        if(m*n != len){
            int [][] ans = new int[0][0];
            return ans;
            // return new int[][]{};
        }

        //we can solve this
        int [][] ans = new int[m][n];

        for(int idx = 0; idx < len; idx++){
            int ele = original[idx];

        //corresponding idx for 2D array
            int x = idx / n;
            int y = idx % n;

            ans[x][y] = ele;

        }

        return ans;
    }
```



Shift Matrix Row-Wise

Sample Input 0

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

n

k =

Sample Output 0

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

```
0   5   9 ✓
2   7   5
4   3   6
```

K=2

```
9 0 5
5 2 7
6 4 3
```

✓

Rotate.

$$\underbrace{1 \quad 2}_{} \quad \underline{3 \quad 4 \quad 5}$$

$\underline{\underline{k=2}}$

↑

$$\boxed{3 \quad 4 \quad 5 \quad 1 \quad 2}$$

$k=2$

$$\underbrace{0 \quad 5}_{} \quad \underline{9}$$

$$\boxed{9 \quad 0 \quad 5}$$

1 2 3 4 5 $k=2 \rightarrow$ 4 5 1 2 3

$r(0, n-k-1)$

$r(n-k, n-1)$

$r(0, n-1)$

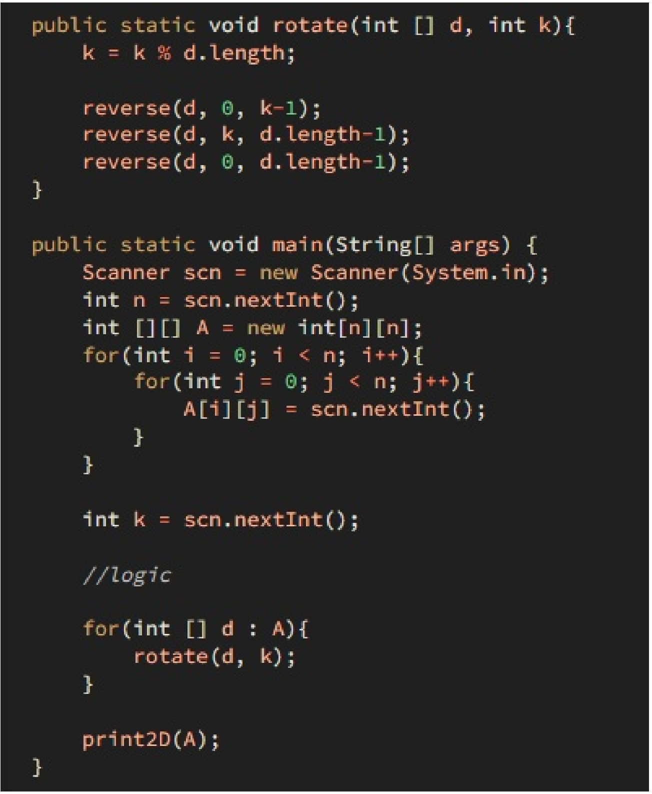
$\underbrace{1\ \ 2}\ \underbrace{3\ \ 4\ \ 5}$ $k=2 \longrightarrow$ 3 4 5 1 2

$r(0, k-1)$

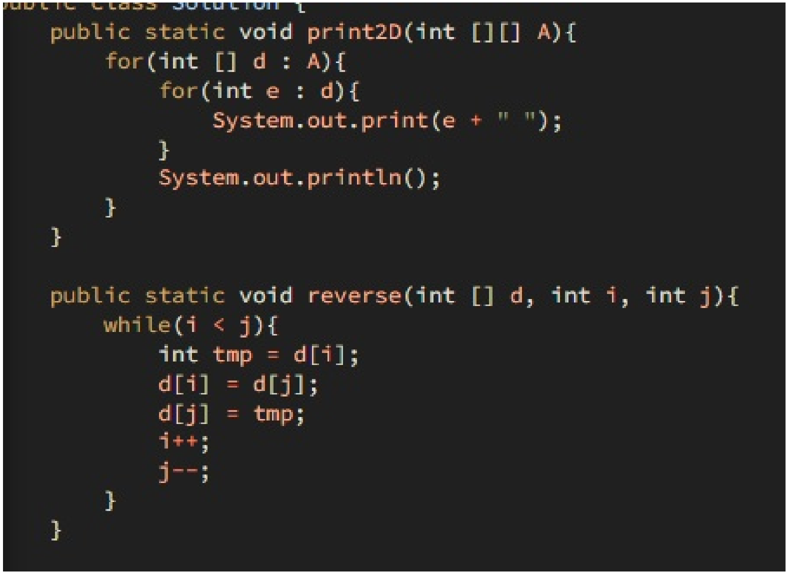
$r(k, n-1)$

$r(0, n-1)$

Shift Matrix Row-Wise



```
public static void rotate(int [] d, int k){
    k = k % d.length;

    reverse(d, 0, k-1);
    reverse(d, k, d.length-1);
    reverse(d, 0, d.length-1);
}

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 k = scn.nextInt();

    //logic

    for(int [] d : A){
        rotate(d, k);
    }

    print2D(A);
}
```

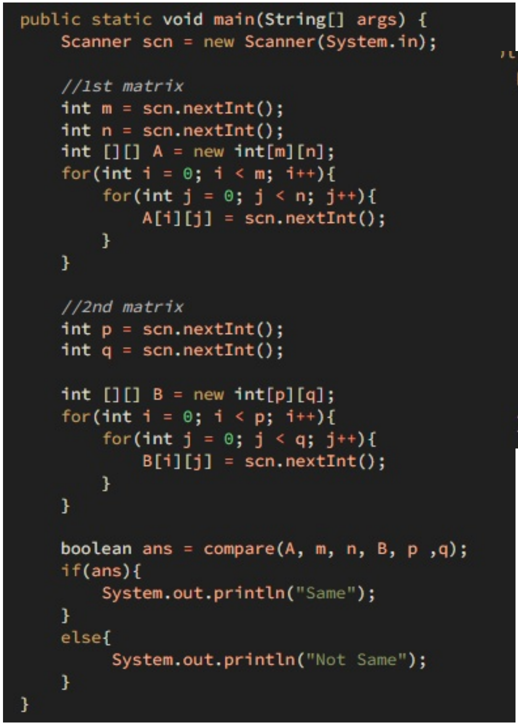


```
public class Solution {
    public static void print2D(int [][] A){
        for(int [] d : A){
            for(int e : d){
                System.out.print(e + " ");
            }
            System.out.println();
        }
    }

    public static void reverse(int [] d, int i, int j){
        while(i < j){
            int tmp = d[i];
            d[i] = d[j];
            d[j] = tmp;
            i++;
            j--;
        }
    }
```

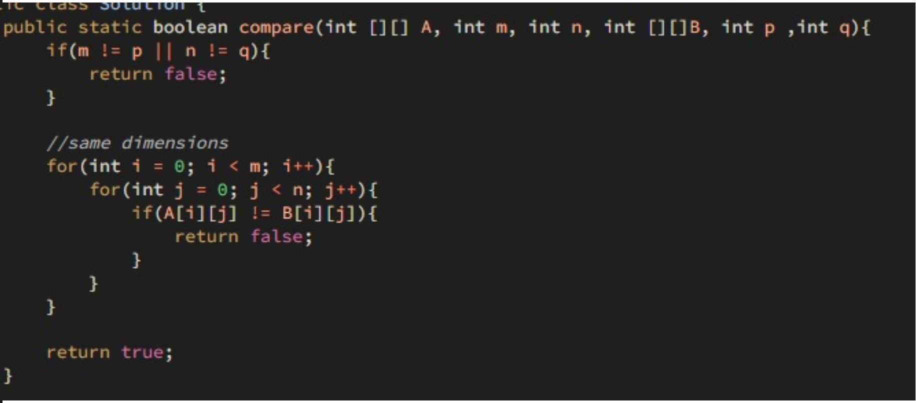
k=0	1	2	3	4	k=4	1	2	3	4
k=1	4	1	2	3	k=5	4	1	2	3
k=2	3	4	1	2	k=6	3	4	1	2
k=3	2	3	4	1					

$k=6 \longrightarrow k=2$

Compare Two Matrices



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

    //1st matrix
    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();
        }
    }

    //2nd matrix
    int p = scn.nextInt();
    int q = scn.nextInt();

    int [][] B = new int[p][q];
    for(int i = 0; i < p; i++){
        for(int j = 0; j < q; j++){
            B[i][j] = scn.nextInt();
        }
    }

    boolean ans = compare(A, m, n, B, p ,q);
    if(ans){
        System.out.println("Same");
    }
    else{
         System.out.println("Not Same");
    }
}
```



```
lic class Solution {
public static boolean compare(int [][] A, int m, int n, int [][]B, int p ,int q){
    if(m != p || n != q){
        return false;
    }

    //same dimensions
    for(int i = 0; i < m; i++){
        for(int j = 0; j < n; j++){
            if(A[i][j] != B[i][j]){
                return false;
            }
        }
    }

    return true;
}
```