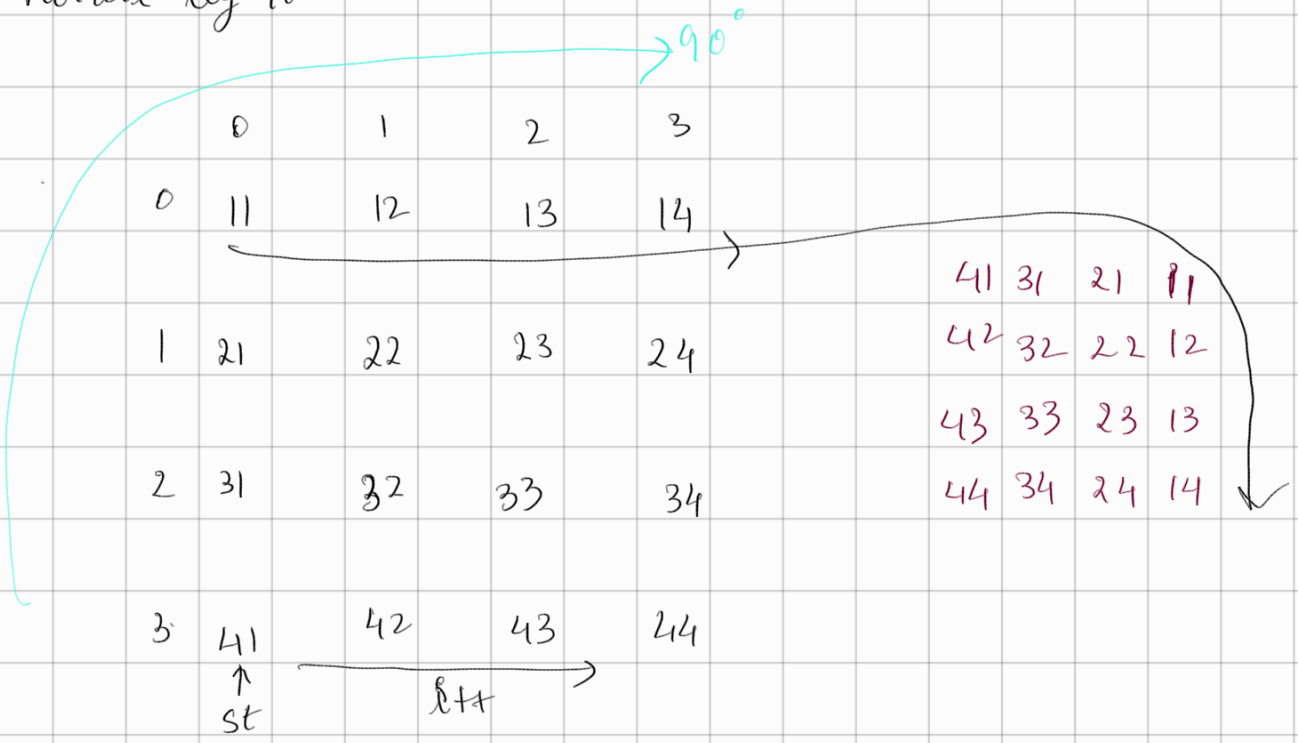
Todays Questions.

① Rotate by 90° ③ Saddle Price

② Ring Rotate (HW) ④ Matrix Multiplication

① Rotate by 90°



first row left to right → last column top to bottom.

Approach

Start from last row & print to first row

↓ bottom to ↓ top

in increasing order columns

```
for (int j=0; j<m; j++) {
    for (int i= n-1; i>=0; i--)
        System.out.print(mat[i][j] + "  ");
    System.out.println("");
}
```

OR

take Transpose.

a[i][j] = a[j][i]

Traverse any one of upper triangle or lower triangle matrix & swap

mat[i][j] & mat[j][i].

Dont swap complete matrix, only any one triangle

diagonal remain same & others are swapped

after transpose, reverse the rows or swap columns in the matrix

① Transpose Code:

```
for(int i=0; i<n ; i++){
    for (int j= i+1; j<n; j++){
        swap (mat, i, j);    // swaps mat[i][j] & mat[j][i]
    }
}
```

② Swap Code:

```
int temp=mat[i][j];
mat[i][j] = mat[j][i];
mat[j][i] = temp;
```

③ Reverse Rows code:

```
for (int i=0; i<n; i++){
    for(int j=0; j<n/2; j++){
        int temp = mat[i][j];
        mat[i][j]= mat[i][n-1-j];
        mat[i][n-1-j] = temp;
    }
}
```

for ①, ② & ③ make functions to make code modular.

④ Swapping columns.

```
int leftCol = 0, rightCol = arr[0].length - 1;
while (leftCol < rightCol) {
    for (int i = 0; i < arr.length; i++) {
        swap(arr, i, leftCol, rightCol);
    }
    leftCol++;
    rightCol--;
}
```

⑤ Overloading swap function for swapping column

```
void swap(int[][] arr, int i, int leftCol, int rightCol) {
    int temp = arr[i][leftCol];
    arr[i][leftCol] = arr[i][rightCol];
    arr[i][rightCol] = temp;
}
```

★ Function with same name but different number or types of arguments are called overloaded functions/methods.

② Matrix multiplication.

$A_{n_1 \times m_1} \times b_{n_2 \times m_2}$ is only possible if $m_1 = n_2$

$A \times B = C$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{bmatrix}_{3\times4} \times \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \\ b_{41} & b_{42} \end{bmatrix}_{4\times2} = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \\ C_{31} & C_{32} \end{bmatrix}_{3\times2}$$

$$c_{11} = a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} + a_{14}b_{41}$$
$$c_{12} = a_{11}b_{12} + a_{12}b_{22} + a_{13}b_{32} + a_{14}b_{42}$$
$$c_{21} = a_{21}b_{11} + a_{22}b_{21} + a_{23}b_{31} + a_{24}b_{41}$$
$$c_{22} = a_{21}b_{12} + a_{22}b_{22} + a_{23}b_{32} + a_{24}b_{42}$$
$$c_{31} = a_{31}b_{11} + a_{32}b_{21} + a_{33}b_{31} + a_{34}b_{41}$$
$$c_{32} = a_{31}b_{12} + a_{32}b_{22} + a_{33}b_{32} + a_{34}b_{42}$$

$$c_{ij} = \sum a_{ik} \times b_{kj}$$

① if (m1 != n2) → Print ("Invalid input"); return;.

approach ①:

make a new matrix → ans [n1][m2]

and loop over for all variables in ans

calculate them & place in ans.

CODE

```
int[][] ans = new int[n1][m2];
for(int i=0; i<m1; i++){
    for (int j=0; j<m2; j++){
        ans[i][j] = 0;
        for (int k=0; k<m1; k++)
            ans[i][j] += mat1[i][k] * mat2[k][j];
    }
}

// print output
for (int i=0; i<n1; i++) {
    for (int j=0; j<m2; j++){
        System.out.print(ans[i][j] + " ");
    }
    System.out.println(" ");
}.
```

Formula: for answer matrix C:

$$C_{ij} = \sum a_{ik} \times b_{kj}$$

③ Saddle Price.

given square matrix, output saddle price

Saddle price → largest in column, smallest in row

all elements are unique

Eg:

34 ✗	42 ✗	30 ✓ 2✗	59 ✗
66 ✗	85 ✗	44 ✓ 2✓	97 ✗
42 ✗	16 ✓ 2✗	25 ✗	36 ✗
15 ✗	21 ✗	05 ✓ 2✗	50 ✗

Saddle Point

(Minimum in row ① & Maximum in column ②)

① Zero saddle point possible?

Yes Eg: Replace 66 with 22 in above matrix

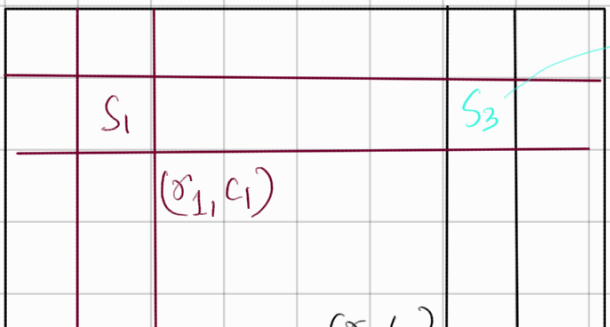
[1][0]

② more than one saddle point

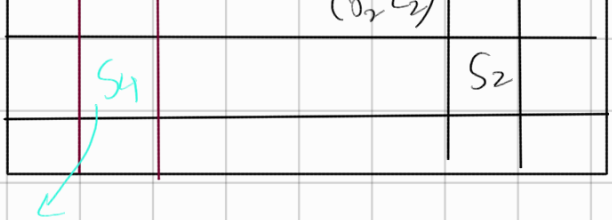
No, Proof below

→ Assume 2 saddle point S_1, S_2

mat



$S_3 > S_1$ & $S_3 < S_2$ ①



$S_4 > S_2$ & $S_4 < S_1$ — (2)

i.e. $S_1 < S_3$ & $S_2 > S_3$, $S_2 < S_4$ & $S_1 > S_4$.

$S_1 < S_3 < S_2$, $S_2 < S_4 < S_1$

$S_1 < S_2$ & $S_2 < S_1$

Cannot be possible because elements are all distinct.

Therefore, there will always be atmost 1 saddle point.

Approach

find minval rowwise for each row

check if minval is maxval for it's column.

if saddle found then print & return.

☆ Math Needed:

*** ① Permutations and Combinations.

** ② Sequence and series. → AP, GP, AGP.

* ③ Number Theory

- Prime numbers
- GCD & LCM
- Factorial

** ④ Logarithm.

⑤ Sets, Relations and Functions.