

# Rajalakshmi Engineering College

Name: Mohammed Rizwan  
Email: 240701327@rajalakshmi.edu.in  
Roll no: 240701327  
Phone: 9944383207  
Branch: REC  
Department: CSE - Section 9  
Batch: 2028  
Degree: B.E - CSE

Scan to verify results



## 2024\_28\_III\_OOPS Using Java Lab

### REC\_2028\_OOPS using Java\_Week 3\_CY

Attempt : 1  
Total Mark : 40  
Marks Obtained : 40

#### Section 1 : Coding

##### 1. Problem Statement:

Emma, a budding computer vision enthusiast, is working on a challenging image processing project. She has a square image represented as a 2D matrix of integers. As part of a special filter operation, she needs to rotate the image by 90 degrees clockwise, but there's a twist – she must perform the rotation in-place, using no extra space.

This means Emma has to rotate the matrix without creating a new one. Your task is to help her implement a Java program that takes this square matrix as input and rotates it within the same structure.

Can you help Emma efficiently rotate the image so that her project can move to the next stage?

***Input Format***

The first line of input contains a single integer  $n$ , representing the number of rows and columns of the square matrix (i.e., the matrix is of size  $n \times n$ ).

The next  $n$  lines each contain  $n$  space-separated integers, representing the elements of each row of the 2D array.

### **Output Format**

The first line of output prints "Rotated 2D Array:"

The next  $n$  lines of output print the rotated matrix.

Each line contains  $n$  space-separated integers representing a row of the rotated matrix.

Refer to the sample output for format specification.

### **Sample Test Case**

Input: 3

1 2 3

4 5 6

7 8 9

Output: Rotated 2D Array:

7 4 1

8 5 2

9 6 3

### **Answer**

// You are using Java

import java.util.\*;

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

```
        int n = sc.nextInt();  
        int[][] matrix = new int[n][n];
```

```
        for (int i = 0; i < n; i++) {  
            for (int j = 0; j < n; j++) {
```

```

        matrix[i][j] = sc.nextInt();
    }
}

rotateMatrix(matrix, n);

System.out.println("Rotated 2D Array:");
for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
        System.out.print(matrix[i][j] + " ");
    }
    System.out.println();
}

public static void rotateMatrix(int[][] matrix, int n) {
    for (int i = 0; i < n; i++) {
        for (int j = i + 1; j < n; j++) {
            int temp = matrix[i][j];
            matrix[i][j] = matrix[j][i];
            matrix[j][i] = temp;
        }
    }
    for (int i = 0; i < n; i++) {
        int left = 0;
        int right = n - 1;
        while (left < right) {
            int temp = matrix[i][left];
            matrix[i][left] = matrix[i][right];
            matrix[i][right] = temp;
            left++;
            right--;
        }
    }
}

```

**Status :** Correct

**Marks :** 10/10

## 2. Problem Statement

Nikila is working as an intern in a software firm and is practicing with a matrix where each row represents a set of numerical values. Her task is to identify the row with the highest sum of its elements and remove that row from the matrix. After removing the row with the highest sum, Nikila needs to print the updated matrix.

Your task is to help Nikila in implementing the same. If there are two or more rows that have same the highest sum, the firstly encountered row is deleted.

### ***Input Format***

The first line of the input consists of two space-separated integers, R and C, representing the number of rows and columns in the matrix, respectively.

The following R lines each contain, C space-separated integers representing the matrix elements.

### ***Output Format***

The output prints the matrix after removing the row with the highest sum. Each row should be printed on a new line, with elements separated by a space.

Refer to the sample output for the formatting specifications.

### ***Sample Test Case***

Input: 2 2

1 2

3 4

Output: 1 2

### ***Answer***

```
// You are using Java
```

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

```
class Main {
```

```
    public static void main(String[] args) {
```

```

Scanner sc = new Scanner(System.in);

int R = sc.nextInt();
int C = sc.nextInt();
int[][] matrix = new int[R][C];

for (int i = 0; i < R; i++) {
    for (int j = 0; j < C; j++) {
        matrix[i][j] = sc.nextInt();
    }
}

int maxSum = Integer.MIN_VALUE;
int rowToRemove = -1;

for (int i = 0; i < R; i++) {
    int rowSum = 0;
    for (int j = 0; j < C; j++) {
        rowSum += matrix[i][j];
    }
    if (rowSum > maxSum) {
        maxSum = rowSum;
        rowToRemove = i;
    }
}

for (int i = 0; i < R; i++) {
    if (i != rowToRemove) {
        for (int j = 0; j < C; j++) {
            System.out.print(matrix[i][j] + " ");
        }
        System.out.println();
    }
}
}

```

**Status :** Correct

**Marks :** 10/10

### 3. Problem Statement

Emma is a data analyst working with a grid-based system where each cell contains important numerical data. The grid represents spatial data, inventory records, or structured reports that require periodic updates.

Due to system updates and new requirements, Emma needs to modify the grid in the following ways:

She wants to insert either a new row or a new column at a given position. Later, she needs to delete either a row or a column from the modified matrix.

#### **Input Format**

The first line contains two integers rows and cols (the dimensions of the matrix).

The next rows lines contain cols space-separated integers representing the initial matrix.

The next line contains two integers insertType and insertIndex:

- insertType = 0 for row insertion, 1 for column insertion.
- insertIndex is the position where the new row/column should be added.

If inserting a row, the next cols integers represent the new row or If inserting a column, the next rows integers represent the new column.

The next line contains two integers deleteType and deleteIndex:

- deleteType = 0 for row deletion, 1 for column deletion.
- deleteIndex is the position to be deleted.

#### **Output Format**

The first line of output prints the string "After insertion" followed by the modified matrix with the inserted row or column.

Each row of the matrix is printed on a new line with space-separated integers.

The next line prints the string "After deletion" followed by the final matrix after the specified deletion operation.

Each row of the resulting matrix is printed on a new line with space-separated integers.

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 3 3

1 2 3

4 5 6

7 8 9

0 1

10 11 12

1 2

Output: After insertion

1 2 3

10 11 12

4 5 6

7 8 9

After deletion

1 2

10 11

4 5

7 8

### **Answer**

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

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

```
  
        int rows = sc.nextInt();  
        int cols = sc.nextInt();  
        int[][] matrix = new int[rows][cols];
```

```
  
        for (int i = 0; i < rows; i++) {  
            for (int j = 0; j < cols; j++) {  
                matrix[i][j] = sc.nextInt();  
            }  
        }
```

```
int insertType = sc.nextInt();
int insertIndex = sc.nextInt();
int[] newRowOrColumn = new int[insertType == 0 ? cols : rows];
```

```
for (int i = 0; i < (insertType == 0 ? cols : rows); i++) {
    newRowOrColumn[i] = sc.nextInt();
}
```

```
if (insertType == 0) {
    matrix = insertRow(matrix, insertIndex, newRowOrColumn);
} else {
    matrix = insertColumn(matrix, insertIndex, newRowOrColumn);
}
```

```
System.out.println("After insertion");
printMatrix(matrix);
```

```
int deleteType = sc.nextInt();
int deleteIndex = sc.nextInt();
```

```
if (deleteType == 0) {
    matrix = deleteRow(matrix, deleteIndex);
} else {
    matrix = deleteColumn(matrix, deleteIndex);
}
```

```
System.out.println("After deletion");
printMatrix(matrix);
}
```

```
public static int[][] insertRow(int[][] matrix, int insertIndex, int[] newRow) {
    int rows = matrix.length;
    int cols = matrix[0].length;
    int[][] newMatrix = new int[rows + 1][cols];
```

```
    for (int i = 0; i < insertIndex; i++) {
        System.arraycopy(matrix[i], 0, newMatrix[i], 0, cols);
    }
```

```
    System.arraycopy(newRow, 0, newMatrix[insertIndex], 0, cols);
```



```

        for (int i = insertIndex; i < rows; i++) {
            System.arraycopy(matrix[i], 0, newMatrix[i + 1], 0, cols);
        }

        return newMatrix;
    }

    public static int[][] insertColumn(int[][] matrix, int insertIndex, int[] newColumn)
    {
        int rows = matrix.length;
        int cols = matrix[0].length;
        int[][] newMatrix = new int[rows][cols + 1];

        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < insertIndex; j++) {
                newMatrix[i][j] = matrix[i][j];
            }
            newMatrix[i][insertIndex] = newColumn[i];
            for (int j = insertIndex; j < cols; j++) {
                newMatrix[i][j + 1] = matrix[i][j];
            }
        }

        return newMatrix;
    }

    public static int[][] deleteRow(int[][] matrix, int deleteIndex) {
        int rows = matrix.length;
        int cols = matrix[0].length;
        int[][] newMatrix = new int[rows - 1][cols];

        for (int i = 0; i < deleteIndex; i++) {
            System.arraycopy(matrix[i], 0, newMatrix[i], 0, cols);
        }

        for (int i = deleteIndex + 1; i < rows; i++) {
            System.arraycopy(matrix[i], 0, newMatrix[i - 1], 0, cols);
        }

        return newMatrix;
    }

```

```

public static int[][] deleteColumn(int[][] matrix, int deleteIndex) {
    int rows = matrix.length;
    int cols = matrix[0].length;
    int[][] newMatrix = new int[rows][cols - 1];

    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < deleteIndex; j++) {
            newMatrix[i][j] = matrix[i][j];
        }
        for (int j = deleteIndex + 1; j < cols; j++) {
            newMatrix[i][j - 1] = matrix[i][j];
        }
    }

    return newMatrix;
}

public static void printMatrix(int[][] matrix) {
    for (int[] row : matrix) {
        for (int cell : row) {
            System.out.print(cell + " ");
        }
        System.out.println();
    }
}

```

**Status :** Correct

**Marks :** 10/10

#### 4. Problem Statement:

Mason is participating in a coding challenge where he must manipulate an integer array. His task is to replace every element in the array with the next greatest element to its right. The last element of the array remains unchanged, as there is no element to its right.

Your job is to help Mason write a program that performs this transformation and outputs the modified array.

### ***Input Format***

The first line of input contains an integer  $n$  representing the number of elements in the array.

The second line of input contains  $n$  space-separated integers representing the elements of the array.

### ***Output Format***

The output prints the modified array of  $n$  integers, where each element (except the last one) is replaced by the maximum element to its right, and the last element remains unchanged.

Refer to the sample output for formatting specifications.

### ***Sample Test Case***

Input: 6

12 3 91 15 12 14

Output: 91 91 15 14 14 14

### ***Answer***

// You are using Java

import java.util.\*;

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

```
        int n = sc.nextInt();  
        int[] arr = new int[n];
```

```
        for (int i = 0; i < n; i++) {  
            arr[i] = sc.nextInt();  
        }
```

```
        int maxSoFar = arr[n - 1];
```

```
        for (int i = n - 2; i >= 0; i--) {  
            int current = arr[i];
```

```
arr[i] = maxSoFar;  
maxSoFar = Math.max(maxSoFar, current);  
}  
  
for (int i = 0; i < n; i++) {  
    System.out.print(arr[i] + " ");  
}  
}  
}
```

**Status :** Correct

**Marks :** 10/10