

# Rajalakshmi Engineering College

Name: Saranya Vimalanathan  
Email: 240701620@rajalakshmi.edu.in  
Roll no: 240701620  
Phone: 9789689339  
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

Alex is a treasure hunter who collects valuable items during their quests. Each item has a specific point value, and Alex wants to maximize their score by strategically removing items one at a time.

The rule is simple: Alex removes the item with the highest point value in each step until no items are left, summing the values of the removed items to calculate the maximum score.

Help Alex to complete his task.

##### ***Input Format***

The first line of input consists of an integer N, representing the size of the array.

The second line of input consists of N space-separated integers, representing the point values of the items.

### **Output Format**

The output prints "Maximum Sum: " followed by the calculated maximum score after removing all items.

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 14

7 14 21 28 35 42 49 56 63 70 77 84 91 98

Output: Maximum Sum: 735

### **Answer**

```
// You are using Java
import java.util.Scanner;
```

```
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int N = sc.nextInt();
        int[] arr = new int[N];
        int sum = 0;
        for (int i = 0; i < N; i++) {
            arr[i] = sc.nextInt();
            sum += arr[i];
        }
        System.out.println("Maximum Sum: " + sum);
        sc.close();
    }
}
```

**Status :** Correct

**Marks :** 10/10

## 2. Problem Statement

Rina is managing the inventory for a library, where each row of a 2D matrix represents the number of different genres of books available on each shelf.

She wants to perform the following operations:

**Transformation:** Replace each element in a row with the sum of all elements in that row.  
**Merging:** After transformation, Rina will provide one additional matrix, and specify whether to merge the transformed matrix with this new matrix row-wise or column-wise.

### ***Input Format***

The first line contains two integers R and C, representing the number of rows and columns of the initial matrix.

The next R lines contain C space-separated integers, representing the book counts in the library.

The next line contains two integers MR and MC, representing the dimensions of the second matrix (to be merged).

The next MR lines contain MC space-separated integers, representing the second matrix.

The last line contains an integer mergeType:

- 0 Row-wise merging (append the second matrix below the transformed matrix).
- 1 Column-wise merging (append the second matrix to the right of the transformed matrix).

### ***Output Format***

The output prints "Transformed matrix: " followed by the transformed 2D matrix where each element in a row is replaced with the sum of the elements in that row.

The output prints "Final merged matrix: ", followed by the merging based on mergeType.

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 3 4

8 2 4 9

4 5 6 1

7 8 9 3

2 4

3 5 7 2

6 1 4 9

0

Output: Transformed matrix:

23 23 23 23

16 16 16 16

27 27 27 27

Final merged matrix:

23 23 23 23

16 16 16 16

27 27 27 27

3 5 7 2

6 1 4 9

### **Answer**

// You are using Java

import java.util.\*;

```
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int R = sc.nextInt(), 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 MR = sc.nextInt(), MC = sc.nextInt();
        int[][] matrix2 = new int[MR][MC];
        for (int i = 0; i < MR; i++)
            for (int j = 0; j < MC; j++)
                matrix2[i][j] = sc.nextInt();
        int mergeType = sc.nextInt();

        int[][] transformed = new int[R][C];
```

```

for (int i = 0; i < R; i++) {
    int rowSum = 0;
    for (int j = 0; j < C; j++) rowSum += matrix[i][j];
    for (int j = 0; j < C; j++) transformed[i][j] = rowSum;
}

System.out.print("Transformed matrix: ");
for (int i = 0; i < R; i++)
    for (int j = 0; j < C; j++)
        System.out.print(transformed[i][j] + " ");
System.out.println();

System.out.print("Final merged matrix: ");
if (mergeType == 0) {
    for (int i = 0; i < R; i++)
        for (int j = 0; j < C; j++)
            System.out.print(transformed[i][j] + " ");
    for (int i = 0; i < MR; i++)
        for (int j = 0; j < MC; j++)
            System.out.print(matrix2[i][j] + " ");
} else {
    for (int i = 0; i < R; i++) {
        for (int j = 0; j < C; j++)
            System.out.print(transformed[i][j] + " ");
        for (int j = 0; j < MC; j++)
            System.out.print(matrix2[i][j] + " ");
    }
}
}
}
}

```

**Status :** Correct

**Marks :** 10/10

### 3. 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.*;
```

```
public 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[] result = new int[n];
        int maxRight = arr[n - 1];
        result[n - 1] = arr[n - 1];

        for (int i = n - 2; i >= 0; i--) {
```

```

        result[i] = maxRight;
        if (arr[i] > maxRight) maxRight = arr[i];
    }

    for (int i = 0; i < n; i++) System.out.print(result[i] + " ");
}
}

```

**Status :** Correct

**Marks :** 10/10

#### 4. 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.\*;

```
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int R = sc.nextInt(), 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, maxRow = -1;
        for (int i = 0; i < R; i++) {
            int sum = 0;
            for (int j = 0; j < C; j++) sum += matrix[i][j];
            if (sum > maxSum) {
                maxSum = sum;
                maxRow = i;
            }
        }

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

**Status :** Correct

**Marks :** 10/10