Rajalakshmi Engineering College

Name: Mohamed Yahya A
Email: 240701325@rajalakshmi.edu.in
Roll no: 240701325
Phone: 9600561844
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.*;
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);
    }
}
```

Status : Correct ***Marks : 10/10***

2. Problem Statement

Robin is a tech-savvy teenager who is diving into programming.

He is working on a project to find special elements in an array called 'leaders.' Leaders are those exceptional elements that are greater than the sum of all the elements to their right.

Assist Robin in writing this program.

Example

Input:

6

16 28 74 19 25 11

Output:

74 25 11

Explanation:

The element 16 is not greater than the sum of elements to its right (28 + 74 + 19 + 25 + 11 = 157)

The element 28 is not greater than the sum of elements to its right (74 + 19 + 25 + 11 = 129)

The element 74 is greater than the sum of elements to its right (19 + 25 + 11 = 55)

The element 19 is not greater than the sum of elements to its right (25 + 11 = 36)

The element 25 is greater than the sum of elements to its right (11)

The last element 11 is always a leader since there are no elements to its right.

So, the output is {74, 25, 11}.

Input Format

The first line of input consists of an integer N, representing the number of elements in the array.

The second line consists of N space-separated integers, representing the elements of the array.

Output Format

The output prints the special elements in the given array, that are greater than the sum of all the elements to their right.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
3 4 2 5 1

Output: 5 1

Answer

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

        for (int i = 0; i < N; i++) {
            int sum = 0;
            for (int j = i + 1; j < N; j++) {
                sum += arr[j];
            }
            if (arr[i] > sum) {
                System.out.print(arr[i] + " ");
            }
        }
    }
}
```

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

```
// You are using Java
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();

        if (insertType == 0) {
            int[] newRow = new int[cols];
            for (int i = 0; i < cols; i++) {
                newRow[i] = sc.nextInt();
            }
            int[][] newMatrix = new int[rows + 1][cols];
            for (int i = 0, k = 0; i < rows + 1; i++) {
                if (i == insertIndex) {
                    newMatrix[i] = newRow;
                } else {
                    newMatrix[i] = matrix[k++];
                }
            }
            matrix = newMatrix;
            rows++;
        } else {
            int[] newCol = new int[rows];
            for (int i = 0; i < rows; i++) {
                newCol[i] = sc.nextInt();
            }
            int[][] newMatrix = new int[rows][cols + 1];
            for (int i = 0; i < rows; i++) {
                for (int j = 0, k = 0; j < cols + 1; j++) {
                    if (j == insertIndex) {
                        newMatrix[i][j] = newCol[i];
                    } else {
                        newMatrix[i][j] = matrix[i][k++];
                    }
                }
            }
            matrix = newMatrix;
            cols++;
        }

        System.out.println("After insertion");
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                System.out.print(matrix[i][j] + " ");
            }
            System.out.println();
        }

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

        if (deleteType == 0) {
            int[][] newMatrix = new int[rows - 1][cols];
            for (int i = 0, k = 0; i < rows; i++) {
                if (i != deleteIndex) {
                    newMatrix[k++] = matrix[i];
                }
            }
            matrix = newMatrix;
            rows--;
        } else {
            int[][] newMatrix = new int[rows][cols - 1];
            for (int i = 0; i < rows; i++) {
                for (int j = 0, k = 0; j < cols; j++) {
                    if (j != deleteIndex) {
                        newMatrix[i][k++] = matrix[i][j];
                    }
                }
            }
            matrix = newMatrix;
            cols--;
        }

        System.out.println("After deletion");
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                System.out.print(matrix[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

Status : Correct ***Marks : 10/10***

4. 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 x 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();
            }
        }

        // Step 1: Transpose the matrix
        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;
            }
        }

        // Step 2: Reverse each row
        for (int i = 0; i < n; i++) {
            int left = 0, right = n - 1;
            while (left < right) {
                int temp = matrix[i][left];
                matrix[i][left] = matrix[i][right];
                matrix[i][right] = temp;
                left++;
                right--;
            }
        }
        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();
        }
    }
}
```

Status : Correct ***Marks : 10/10***