Rajalakshmi Engineering College

Name: Samyuktha S
Email: 240701701@rajalakshmi.edu.in
Roll no: 240701701
Phone: 6380226314
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

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

```
import java.util.*;

public 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) continue;
            for (int j = 0; j < C; j++) {
                System.out.print(matrix[i][j]);
                if (j < C - 1)
                    System.out.print(" ");
            }
            System.out.println();
        }
    }
}
```

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

2. 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 MatrixEditor {
   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; i < insertIndex; i++)
            newMatrix[i] = matrix[i];
         newMatrix[insertIndex] = newRow;
         for (int i = insertIndex; i < rows; i++)
            newMatrix[i + 1] = matrix[i];
         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; j < insertIndex; j++)
               newMatrix[i][j] = matrix[i][j];
            newMatrix[i][insertIndex] = newCol[i];
            for (int j = insertIndex; j < cols; j++)
```

```
            newMatrix[i][j + 1] = matrix[i][j];
        }
        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) continue;
            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) continue;
                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**

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

```
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 max = arr[n - 1];
        for(int i = n - 2; i >= 0; i--) {
            int temp = arr[i];
            arr[i] = max;
            if(temp > max) {
                max = temp;
            }
        }

        for(int i = 0; i < n; i++) {
            System.out.print(arr[i]);
            if(i != n - 1) System.out.print(" ");
        }
    }
}
```

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

4. Problem Statement:

Imagine you have an array of integer values, and you're tasked with identifying a pair of elements within the array. This pair of elements should have a sum that is the closest to zero when compared to any other pair in the array.

Your goal is to create a program that solves this problem efficiently. The program should accept an array of integers and return the pair of elements whose sum is closest to zero.

Input Format

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

The second line of the input contains N space-separated integer values.

Output Format

The output is displayed in the following format:

"Pair with the sum closest to zero: {value} and {value}"

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
9 10 -3 -5 -2
Output: Pair with the sum closest to zero: 9 and -5

Answer

```
import java.util.*;

class ClosestToZero {
   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 minSum = Integer.MAX_VALUE;
      int first = 0, second = 0;

      for(int i = 0; i < n - 1; i++) {
         for(int j = i + 1; j < n; j++) {
            int sum = arr[i] + arr[j];
            if(Math.abs(sum) < Math.abs(minSum)) {
               minSum = sum;
               first = arr[i];
```

```
                second = arr[j];
            }
        }
    }

    System.out.println("Pair with the sum closest to zero: " + first + " and " +
second);
  }
}
```

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