Rajalakshmi Engineering College

Name: Jathin Chowdary CH
Email: 240701209@rajalakshmi.edu.in
Roll no: 240701209
Phone: 8939767767
Branch: REC
Department: CSE - Section 10
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.Scanner;

class MatrixRowRemoval {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

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

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

        int maxSumRowIndex = 0;
        int maxRowSum = Integer.MIN_VALUE;

        for (int i = 0; i < rows; i++) {
            int rowSum = 0;
            for (int j = 0; j < cols; j++) {
```

```
            rowSum += matrix[i][j];
        }
        if (rowSum > maxRowSum) {
            maxRowSum = rowSum;
            maxSumRowIndex = i;
        }
    }

    int[][] newMatrix = new int[rows - 1][cols];
    int newRowIndex = 0;

    for (int i = 0; i < rows; i++) {
        if (i == maxSumRowIndex) {
            continue;
        }
        for (int j = 0; j < cols; j++) {
            newMatrix[newRowIndex][j] = matrix[i][j];
        }
        newRowIndex++;
    }

    for (int i = 0; i < rows - 1; i++) {
        for (int j = 0; j < cols; j++) {
            System.out.print(newMatrix[i][j] + " ");
        }
        System.out.println();
    }

    scanner.close();
  }
}
```

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

2. 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

```
import java.util.Scanner;
class MaxSumReduction {
public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        int size = scanner.nextInt();
        int[] arr = new int[size];

        for (int i = 0; i < size; i++) {
            arr[i] = scanner.nextInt();
        }

        int maxSum = 0;
```

```
        while (size > 0) {
            int maxIndex = 0;
            int maxElement = arr[0];

            for (int i = 1; i < size; i++) {
                if (arr[i] > maxElement) {
                    maxElement = arr[i];
                    maxIndex = i;
                }
            }

            maxSum += maxElement;

            for (int i = maxIndex; i < size - 1; i++) {
                arr[i] = arr[i + 1];
            }
            size--;
        }

        System.out.println("Maximum Sum: " + maxSum);
    scanner.close();
}
}
```

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

3. 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 ClosestSumToZero {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int n = scanner.nextInt();
        int[] arr = new int[n];
        for (int i = 0; i < n; i++) {
            arr[i] = scanner.nextInt();
        }
        findClosestSumPair(arr, n);

        scanner.close();
    }

    public static void findClosestSumPair(int[] arr, int n) {
        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**

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

```
import java.util.Scanner;

 class InPlaceMatrixRotation {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);

      int n = scanner.nextInt();

      int[][] matrix = new int[n][n];

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

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

      rotateMatrix(matrix);

      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();
}

    scanner.close();
    }

    public static void rotateMatrix(int[][] matrix) {
        int n = matrix.length;

        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++) {
            for (int j = 0; j < n / 2; j++) {

                int temp = matrix[i][j];
                matrix[i][j] = matrix[i][n - j - 1];
                matrix[i][n - j - 1] = temp;
            }
        }
    }
}
```

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