Rajalakshmi Engineering College

Name: Mohit Jha
Email: 241901056@rajalakshmi.edu.in
Roll no:
Phone: 9445934493
Branch: REC
Department: CSE (CS) - Section 1
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

2024_28_III_OOPS Using Java Lab

REC_2028_OOPS using Java_Week 3_CY

Attempt : 2
Total Mark : 40
Marks Obtained : 40

Section 1 : Coding

1. 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.Scanner;

class NextGreatestElement {
    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();
        }

        for (int i = 0; i < n - 1; i++) {
            int max = arr[i + 1];
            for (int j = i + 1; j < n; j++) {
                if (arr[j] > max) {
                    max = arr[j];
                }
            }
            arr[i] = max;
        }

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

```
            System.out.print(" ");
         }
      }
   }
}
```

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

```
import java.util.Scanner;

public class Main {
    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();
        }

        for (int i = 0; i < size; i++) {
            int sumRight = 0;

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

            if (arr[i] > sumRight) {
                System.out.print(arr[i] + " ");
            }
        }
    }
}
```

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

3. 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

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

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

        // Read the initial matrix dimensions (R x C)
        int R = sc.nextInt();
        int C = sc.nextInt();
        int[][] libraryMatrix = new int[R][C];

        // Read the library matrix values
        for (int i = 0; i < R; i++) {
            for (int j = 0; j < C; j++) {
                libraryMatrix[i][j] = sc.nextInt();
            }
        }

        // Apply the transformation (sum of each row)
        int[][] transformedMatrix = new int[R][C];
        for (int i = 0; i < R; i++) {
            int rowSum = 0;
            for (int j = 0; j < C; j++) {
                rowSum += libraryMatrix[i][j];
            }
            // Replace each element in the row with the row sum
            Arrays.fill(transformedMatrix[i], rowSum);
        }

        // Print the transformed matrix
        System.out.println("Transformed matrix:");
        printMatrix(transformedMatrix);

        // Read the second matrix dimensions (MR x MC)
        int MR = sc.nextInt();
        int MC = sc.nextInt();
        int[][] secondMatrix = new int[MR][MC];

        // Read the second matrix values
        for (int i = 0; i < MR; i++) {
            for (int j = 0; j < MC; j++) {
                secondMatrix[i][j] = sc.nextInt();
            }
        }

        // Read the merge type (0 for row-wise, 1 for column-wise)
        int mergeType = sc.nextInt();

        // Perform the merge operation based on the merge type
        int[][] mergedMatrix = null;

        if (mergeType == 0) {
            // Row-wise merging: Append the second matrix below the transformed
matrix
            if (R + MR <= 10) { // Ensure the resulting row count does not exceed the
limit
                mergedMatrix = new int[R + MR][Math.max(C, MC)];

                // Copy the transformed matrix
                for (int i = 0; i < R; i++) {
                    for (int j = 0; j < C; j++) {
                        mergedMatrix[i][j] = transformedMatrix[i][j];
                    }
                }

                // Copy the second matrix below the transformed matrix
                for (int i = 0; i < MR; i++) {
                    for (int j = 0; j < MC; j++) {
                        mergedMatrix[R + i][j] = secondMatrix[i][j];
                    }
                }
            }
        } else if (mergeType == 1) {
            // Column-wise merging: Append the second matrix to the right of the
transformed matrix
            if (C + MC <= 10) { // Ensure the resulting column count does not exceed
the limit
                mergedMatrix = new int[R][C + MC];

                // Copy the transformed matrix
                for (int i = 0; i < R; i++) {
                    for (int j = 0; j < C; j++) {
                        mergedMatrix[i][j] = transformedMatrix[i][j];
                    }
                }
                // Copy the second matrix to the right of the transformed matrix
                for (int i = 0; i < R; i++) {
                    for (int j = 0; j < MC; j++) {
                        mergedMatrix[i][C + j] = secondMatrix[i][j];
                    }
                }
            }
        }

        // Print the final merged matrix
        System.out.println("Final merged matrix:");
        printMatrix(mergedMatrix);

        sc.close();
      }

      // Helper method to print the matrix
      private static void printMatrix(int[][] matrix) {
        for (int i = 0; i < matrix.length; i++) {
            for (int j = 0; j < matrix[i].length; j++) {
                System.out.print(matrix[i][j]);
                if (j < matrix[i].length - 1) {
                    System.out.print(" ");
                }
            }
            System.out.println();
        }
      }
}
```

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

      // Read array size
      int n = scanner.nextInt();

      // Read array elements
      int[] arr = new int[n];
      for (int i = 0; i < n; i++) {
         arr[i] = scanner.nextInt();
      }

      // Find pair with sum closest to zero
```

```
        findClosestSumPair(arr, n);

        scanner.close();
    }

    public static void findClosestSumPair(int[] arr, int n) {
        int minSum = Integer.MAX_VALUE;
        int first = 0, second = 0;

        // Check all possible pairs
        for (int i = 0; i < n - 1; i++) {
            for (int j = i + 1; j < n; j++) {
                int sum = arr[i] + arr[j];

                // Check if this sum is closer to zero
                if (Math.abs(sum) < Math.abs(minSum)) {
                    minSum = sum;
                    first = arr[i];
                    second = arr[j];
                }
            }
        }

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

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