Rajalakshmi Engineering College

Name: Arilli Jagadeesh A
Email: 241901502@rajalakshmi.edu.in
Roll no: 241901502
Phone: 9361250488
Branch: REC
Department: CSE (CS) - Section 2
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 : 1
Total Mark : 40
Marks Obtained : 40

Section 1 : Coding

1. 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.*;

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

        // Read size of array
        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 val1 = 0, val2 = 0;

        // Check all pairs
        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;
                    val1 = arr[i];
                    val2 = arr[j];
                }
            }
        }

        // Print result
        System.out.println("Pair with the sum closest to zero: " + val1 + " and " +
val2);
    }
}
```

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

```
import java.util.*;

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

```
// First matrix input
int R = sc.nextInt();
int C = sc.nextInt();
int[][] mat1 = new int[R][C];
for (int i = 0; i < R; i++) {
    for (int j = 0; j < C; j++) {
        mat1[i][j] = sc.nextInt();
    }
}

// Transformation: replace each element with row sum
int[][] transformed = new int[R][C];
for (int i = 0; i < R; i++) {
    int rowSum = 0;
    for (int j = 0; j < C; j++) {
        rowSum += mat1[i][j];
    }
    for (int j = 0; j < C; j++) {
        transformed[i][j] = rowSum;
    }
}

// Second matrix input
int MR = sc.nextInt();
int MC = sc.nextInt();
int[][] mat2 = new int[MR][MC];
for (int i = 0; i < MR; i++) {
    for (int j = 0; j < MC; j++) {
        mat2[i][j] = sc.nextInt();
    }
}

// Merge type input
int mergeType = sc.nextInt();

// Print transformed matrix
System.out.println("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();
        }

        // Print final merged matrix
        System.out.println("Final merged matrix:");
        if (mergeType == 0) { // Row-wise merge
            // Print transformed first
            for (int i = 0; i < R; i++) {
                for (int j = 0; j < C; j++) {
                    System.out.print(transformed[i][j] + " ");
                }
                System.out.println();
            }
            // Then second matrix
            for (int i = 0; i < MR; i++) {
                for (int j = 0; j < MC; j++) {
                    System.out.print(mat2[i][j] + " ");
                }
                System.out.println();
            }
        } else { // Column-wise merge
            for (int i = 0; i < Math.max(R, MR); i++) {
                if (i < R) {
                    for (int j = 0; j < C; j++) {
                        System.out.print(transformed[i][j] + " ");
                    }
                }
                if (i < MR) {
                    for (int j = 0; j < MC; j++) {
                        System.out.print(mat2[i][j] + " ");
                    }
                }
                System.out.println();
            }
        }
    }
}
```

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

3. 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.*;

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

        // Read size
        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];
        }

        // Print result
        System.out.println("Maximum Sum: " + sum);
    }
}
```

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

4. 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.*;

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();
        }

        List<Integer> leaders = new ArrayList<>();

        // Traverse from left to right
        for (int i = 0; i < N; i++) {
            int rightSum = 0;
            for (int j = i + 1; j < N; j++) {
                rightSum += arr[j];
            }
            if (arr[i] > rightSum) {
                leaders.add(arr[i]);
            }
        }

        // Print leaders
        for (int i = 0; i < leaders.size(); i++) {
            System.out.print(leaders.get(i) + " ");
        }
    }
}
```

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