Rajalakshmi Engineering College

Name: DARSHAN M
Email: 241501040@rajalakshmi.edu.in
Roll no: 241501040
Phone: 9360697087
Branch: REC
Department: AI & ML - Section 1
Batch: 2028
Degree: B.E - AI & ML

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.*;
public class Main {
   public static void main(String [] args){
      Scanner s = new Scanner(System.in);
      int sum = 0;
      int n =s.nextInt();
      int a[]= new int [n];
      for(int i=0;i<n;i++){
         a[i]=s.nextInt();
         sum+=a[i];
      }
      System.out.println("Maximum Sum: "+sum);
   }
}
```

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 {

    // Function to transform matrix (replace row with its sum)
    public static int[][] transformMatrix(int[][] matrix, int R, int C) {
        int[][] transformed = new int[R][C];
        for (int i = 0; i < R; i++) {
            int rowSum = 0;
            for (int j = 0; j < C; j++) {
                rowSum += matrix[i][j];
            }
            for (int j = 0; j < C; j++) {
                transformed[i][j] = rowSum;
            }
        }
        return transformed;
    }

    // Function to print matrix
    public static void printMatrix(int[][] matrix) {
        for (int[] row : matrix) {
```

```
            for (int val : row) {
                System.out.print(val + " ");
            }
            System.out.println();
        }
    }

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

        // Input first matrix
        int R = sc.nextInt();
        int C = sc.nextInt();
        int[][] matrix1 = new int[R][C];
        for (int i = 0; i < R; i++) {
            for (int j = 0; j < C; j++) {
                matrix1[i][j] = sc.nextInt();
            }
        }

        // Input second matrix
        int MR = sc.nextInt();
        int MC = sc.nextInt();
        int[][] matrix2 = new int[MR][MC];
        for (int i = 0; i < MR; i++) {
            for (int j = 0; j < MC; j++) {
                matrix2[i][j] = sc.nextInt();
            }
        }

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

        // Step 1: Transform
        int[][] transformed = transformMatrix(matrix1, R, C);

        System.out.println("Transformed matrix:");
        printMatrix(transformed);

        // Step 2: Merge
        System.out.println("Final merged matrix:");
        if (mergeType == 0) { // Row-wise merge
```

```
            for (int i = 0; i < R; i++) {
                for (int j = 0; j < C; j++) {
                    System.out.print(transformed[i][j] + " ");
                }
                System.out.println();
            }
            for (int i = 0; i < MR; i++) {
                for (int j = 0; j < MC; j++) {
                    System.out.print(matrix2[i][j] + " ");
                }
                System.out.println();
            }
        } else { // Column-wise merge
            for (int i = 0; i < R; i++) {
                for (int j = 0; j < C; j++) {
                    System.out.print(transformed[i][j] + " ");
                }
                for (int j = 0; j < MC; j++) {
                    System.out.print(matrix2[i][j] + " ");
                }
                System.out.println();
            }
        }

        sc.close();
    }
}
```

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

3. 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;

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

      // Step 1: Read rows and columns
      int R = sc.nextInt();
      int C = sc.nextInt();

      int[][] matrix = new int[R][C];

      // Step 2: Input matrix
      for (int i = 0; i < R; i++) {
         for (int j = 0; j < C; j++) {
```

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

    // Step 3: Find row with maximum sum
    int maxSum = Integer.MIN_VALUE;
    int maxRowIndex = -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;
            maxRowIndex = i;  // Store row index
        }
    }

    // Step 4: Print matrix excluding the max row
    for (int i = 0; i < R; i++) {
        if (i == maxRowIndex) continue;  // Skip the row with max sum
        for (int j = 0; j < C; j++) {
            System.out.print(matrix[i][j] + " ");
        }
        System.out.println();
    }

    sc.close();
  }
}
```

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

4. 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.*;

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

        // Step 1: Read matrix
        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();
            }
        }

        // Step 2: Read insertion info
        int insertType = sc.nextInt();
        int insertIndex = sc.nextInt();
```

```
if (insertType == 0) { // Insert row
    int[] newRow = new int[cols];
    for (int j = 0; j < cols; j++) {
        newRow[j] = 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 { // Insert column
    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++;
}

// Print after insertion
System.out.println("After insertion");
printMatrix(matrix, rows, cols);
```

```
        // Step 3: Read deletion info
        int deleteType = sc.nextInt();
        int deleteIndex = sc.nextInt();

        if (deleteType == 0) { // Delete row
            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 { // Delete column
            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--;
        }

        // Print after deletion
        System.out.println("After deletion");
        printMatrix(matrix, rows, cols);

        sc.close();
    }

    // Utility to print matrix
    public static void printMatrix(int[][] mat, int r, int c) {
        for (int i = 0; i < r; i++) {
            for (int j = 0; j < c; j++) {
                System.out.print(mat[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

Status : Correct *Marks : 10/10*