Rajalakshmi Engineering College

Name: SUBASH R
Email: 240701538@rajalakshmi.edu.in
Roll no: 240701538
Phone: 9150202710
Branch: REC
Department: CSE - Section 7
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

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.Scanner;
class MatrixModification {
   public static void main(String[] args) {
      Scanner in = new Scanner(System.in);
      int r = in.nextInt();
      int c = in.nextInt();
      int[][] mat = new int[r][c];
      for (int i = 0; i < r; i++)
         for (int j = 0; j < c; j++)
            mat[i][j] = in.nextInt();
      int inType = in.nextInt();
      int inIdx = in.nextInt();
      int[][] midMat;
      if (inType == 0) {
         midMat = new int[r + 1][c];
         for (int i = 0, ni = 0; i <= r; i++, ni++) {
            if (i == inIdx) {
               for (int j = 0; j < c; j++)
                  midMat[i][j] = in.nextInt();
               ni--;
            } else
               for (int j = 0; j < c; j++)
                  midMat[i][j] = mat[ni][j];
         }
         r++;
      } else {
         midMat = new int[r][c + 1];
         for (int i = 0; i < r; i++)
            for (int j = 0, nj = 0; j <= c; j++, nj++) {
               if (j == inIdx) {
                  midMat[i][j] = in.nextInt();
```

```
                nj--;
            } else
                midMat[i][j] = mat[i][nj];
        }
    c++;
}
System.out.println("After insertion");
for (int i = 0; i < r; i++) {
    for (int j = 0; j < c; j++)
        System.out.print(midMat[i][j] + " ");
    System.out.println();
}
int delType = in.nextInt();
int delIdx = in.nextInt();
int[][] finalMat;
if (delType == 0) {
    finalMat = new int[r - 1][c];
    for (int i = 0, ni = 0; i < r; i++) {
        if (i == delIdx) continue;
        for (int j = 0; j < c; j++)
            finalMat[ni][j] = midMat[i][j];
        ni++;
    }
    r--;
} else {
    finalMat = new int[r][c - 1];
    for (int i = 0; i < r; i++)
        for (int j = 0, nj = 0; j < c; j++) {
            if (j == delIdx) continue;
            finalMat[i][nj++] = midMat[i][j];
        }
    c--;
}
System.out.println("After deletion");
for (int i = 0; i < r; i++) {
    for (int j = 0; j < c; j++)
        System.out.print(finalMat[i][j] + " ");
    System.out.println();
}
in.close();
}
}
```

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

2. 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 in = new Scanner(System.in);
        int r = in.nextInt();
        int c = in.nextInt();
        int[][] mat = new int[r][c];
        for (int i = 0; i < r; i++)
            for (int j = 0; j < c; j++)
                mat[i][j] = in.nextInt();
        int maxIdx = 0;
        int maxSum = Integer.MIN_VALUE;
        for (int i = 0; i < r; i++) {
            int sum = 0;
            for (int j = 0; j < c; j++)
                sum += mat[i][j];
            if (sum > maxSum) {
                maxSum = sum;
                maxIdx = i;
            }
        }
        int[][] newMat = new int[r - 1][c];
        int nr = 0;
        for (int i = 0; i < r; i++) {
            if (i == maxIdx) continue;
            for (int j = 0; j < c; j++)
                newMat[nr][j] = mat[i][j];
            nr++;
        }
        for (int i = 0; i < r - 1; i++) {
            for (int j = 0; j < c; j++)
                System.out.print(newMat[i][j] + " ");
            System.out.println();
        }
        in.close();
    }
}
```

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

3. 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.*;

public class Main{
   public static void main(String args[]){
      Scanner in=new Scanner(System.in);
      int n=in.nextInt();
      int arr[][]=new int[n][n];
      for(int i=0;i<n;i++) for(int j=0;j<n;j++) arr[i][j]=in.nextInt();
      System.out.println("Rotated 2D Array:");
      for(int i=0;i<n;i++){
         for(int j=n-1;j>=0;j--){
            System.out.print(arr[j][i]+" ");
         }
         System.out.println();
      }
   }
}
```

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

4. 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 in = new Scanner(System.in);
      int r1 = in.nextInt();
      int c1 = in.nextInt();
      int[][] mat1 = new int[r1][c1];
      for (int i = 0; i < r1; i++) for (int j = 0; j < c1; j++) mat1[i][j] = in.nextInt();
      int[][] transMat = new int[r1][c1];
      for (int i = 0; i < r1; i++) {
         int rowSum = 0;
         for (int j = 0; j < c1; j++) {
            rowSum += mat1[i][j];
         }
         Arrays.fill(transMat[i], rowSum);
      }
      System.out.println("Transformed matrix:");
      for (int i = 0; i < transMat.length; i++) {
         for (int j = 0; j < transMat[i].length; j++) {
            System.out.print(transMat[i][j]);
            if (j < transMat[i].length - 1) {
               System.out.print(" ");
            }
         }
         System.out.println();
      }
      int r2 = in.nextInt();
      int c2 = in.nextInt();
      int[][] mat2 = new int[r2][c2];
      for (int i = 0; i < r2; i++) for (int j = 0; j < c2; j++) mat2[i][j] = in.nextInt();
      int type = in.nextInt();
      int[][] finalMat = null;
      if (type == 0) {
         if (r1 + r2 <= 10) {
```

```
            finalMat = new int[r1 + r2][Math.max(c1, c2)];
            for (int i = 0; i < r1; i++) {
                for (int j = 0; j < c1; j++) {
                    finalMat[i][j] = transMat[i][j];
                }
            }
            for (int i = 0; i < r2; i++) {
                for (int j = 0; j < c2; j++) {
                    finalMat[r1 + i][j] = mat2[i][j];
                }
            }
        }
    } else if (type == 1) {
        if (c1 + c2 <= 10) {
            finalMat = new int[r1][c1 + c2];
            for (int i = 0; i < r1; i++) {
                for (int j = 0; j < c1; j++) {
                    finalMat[i][j] = transMat[i][j];
                }
            }
            for (int i = 0; i < r1; i++) {
                for (int j = 0; j < c2; j++) {
                    finalMat[i][c1 + j] = mat2[i][j];
                }
            }
        }
    }
    System.out.println("Final merged matrix:");
    for (int i = 0; i < finalMat.length; i++) {
        for (int j = 0; j < finalMat[i].length; j++) {
            System.out.print(finalMat[i][j]);
            if (j < finalMat[i].length - 1) {
                System.out.print(" ");
            }
        }
        System.out.println();
    }
  }
}
```

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