## Matrix Traversal and Manipulation Problems

The following code snippets represent solutions to various matrix traversal and manipulation problems, as provided by Prof. Pradep Kumar from "16\_Sept.pdf".

### Problem 1: Wave Form Traversal

**Description:** This program traverses a given matrix in a wave-like pattern. For columns with even indices (0, 2, 4, ...), the traversal is downwards from top to bottom. For columns with odd indices (1, 3, 5, ...), the traversal is upwards from bottom to top.

import java.util.Scanner;  
public class Wave\_Form\_Traversal {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.in);  
  
 int N = scanner.nextInt(); //Number of rows  
 int M = scanner.nextInt(); //Number of columns  
  
 int[][] matrix = new int[N][M];  
 for (int i = 0; i < matrix.length; i++) {  
 for (int j = 0; j < matrix[i].length; j++) {  
 matrix[i][j] = scanner.nextInt();  
 }  
 }  
 scanner.close();  
 waveFormTraversal(matrix);  
 }  
 public static void waveFormTraversal(int[][] matrix) {  
 for (int j = 0; j < matrix[0].length; j++) {  
 if (j % 2 == 0) {  
 // Traverse downwards for even indexed columns  
 for (int i = 0; i < matrix.length; i++) {  
 System.out.print(matrix[i][j] + " ");  
 }  
 } else {  
 // Traverse upwards for odd indexed columns  
 for (int i = matrix.length - 1; i >= 0; i--) {  
 System.out.print(matrix[i][j] + " ");  
 }  
 }  
 }  
 }  
}

### Problem 2: Transpose Of Matrix

**Description:** This program takes a matrix as input and computes its transpose. The transpose of a matrix is obtained by swapping its rows and columns. The dimensions of the transposed matrix will be M x N if the original matrix was N x M.

import java.util.Scanner;  
public class Transpose\_Of\_Matrix {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.in);  
  
 int N = scanner.nextInt(); //Number of rows  
 int M = scanner.nextInt(); //Number of columns  
  
 int[][] matrix = new int[N][M];  
 for (int i = 0; i < matrix.length; i++) {  
 for (int j = 0; j < matrix[i].length; j++) {  
 matrix[i][j] = scanner.nextInt();  
 }  
 }  
 scanner.close();  
 transposeOfMatrix(matrix);  
 }  
  
 public static void transposeOfMatrix(int[][] matrix) {  
 int[][] transposedMatrix = new int[matrix[0].length][matrix.length];  
  
 for (int i = 0; i < matrix.length; i++) {  
 for (int j = 0; j < matrix[i].length; j++) {  
 transposedMatrix[j][i] = matrix[i][j];  
 }  
 }  
  
 for (int i = 0; i < transposedMatrix.length; i++) {  
 for (int j = 0; j < transposedMatrix[i].length; j++) {  
 System.out.print(transposedMatrix[i][j] + " ");  
 }  
 System.out.println();  
 }  
 }  
}

### Problem 3: Spiral Traversal Of Matrix

**Description:** This program performs a spiral traversal of a given matrix. It starts from the top-left element and moves in a clockwise spiral pattern, printing each element encountered.

import java.util.Scanner;  
public class Spiral\_Traversal\_Of\_Matrix {  
  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.in);  
  
 int N = scanner.nextInt(); //Number of rows  
 int M = scanner.nextInt(); //Number of columns  
  
 int[][] matrix = new int[N][M];  
 for (int i = 0; i < matrix.length; i++) {  
 for (int j = 0; j < matrix[i].length; j++) {  
 matrix[i][j] = scanner.nextInt();  
 }  
 }  
 scanner.close();  
  
 spiralTraversalOfMatrix(matrix);  
 }  
 public static void spiralTraversalOfMatrix(int[][] matrix) {  
 int top = 0, bottom = matrix.length - 1;  
 int left = 0, right = matrix[0].length - 1;  
 while (top <= bottom && left <= right) {  
 for (int i = left; i <= right; i++) {  
 System.out.print(matrix[top][i] + " ");  
 }  
 top++;  
 for (int i = top; i <= bottom; i++) {  
 System.out.print(matrix[i][right] + " ");  
 }  
 right--;  
 if (top <= bottom) {  
 for (int i = right; i >= left; i--) {  
 System.out.print(matrix[bottom][i] + " ");  
 }  
 bottom--;  
 }  
 if (left <= right) {  
 for (int i = bottom; i >= top; i--) {  
 System.out.print(matrix[i][left] + " ");  
 }  
 left++;  
 }  
 }  
 }  
}

### Problem 4: Rotate Clockwise 90deg

**Description:** This program rotates a square matrix 90 degrees clockwise. The logic involves two main steps: first, transposing the matrix, and second, reversing each row of the transposed matrix.

// This problem of rotating 90 degrees clockwise is similar to finding the transpose of a matrix.  
// The only difference is that after finding the transpose of the matrix, we need to reverse each row.  
// For example, if the input matrix is:  
// 1 2 3  
// 4 5 6  
// 7 8 9  
// The transpose of the matrix is:  
// 1 4 7  
// 2 5 8  
// 3 6 9  
// After reversing each row, we get the final output as:  
// 7 4 1  
// 8 5 2  
// 9 6 3  
// Which is the original matrix rotated 90 degrees clockwise.  
  
import java.util.Scanner;  
public class Rotate\_Clockwise\_90deg {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.in);  
  
 int N = scanner.nextInt(); //Number of rows = Number of Columns  
  
 int[][] matrix = new int[N][N];  
 for (int i = 0; i < matrix.length; i++) {  
 for (int j = 0; j < matrix[i].length; j++) {  
 matrix[i][j] = scanner.nextInt();  
 }  
 }  
 scanner.close();  
 rotateClockwise90Deg(matrix);  
 }  
  
 public static void rotateClockwise90Deg(int[][] matrix) {  
 //int row = matrix.length;  
 int N = matrix.length;  
   
 // Step 1: Transpose the matrix  
 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;  
 }  
 }  
  
 // Step 2: Reverse each row  
 for (int i = 0; i < N; i++) {  
 for (int j = 0, k = N - 1; j < k; j++, k--) {  
 int temp = matrix[i][j];  
 matrix[i][j] = matrix[i][k];  
 matrix[i][k] = temp;  
 }  
 }  
  
   
 for (int i = 0; i < N; i++) {  
 for (int j = 0; j < N; j++) {  
 System.out.print(matrix[i][j] + " ");  
 }  
 }  
 }  
}