***SET A:***

**public class MatrixRotation** {

**public static void rotate(int[][] matrix)** {

int n = matrix.length;

//Transpose

for (int i = 0; i < n; i++) {

for (int j = i; j < n; j++) {

// Swap the elements at indices (i, j) and (j, i)

int temp = matrix[i][j];

matrix[i][j] = matrix[j][i];

matrix[j][i] = temp;

}

}

//Reverse

for (int j = 0; j < n; j++) {

reverseColumn(matrix, j);

}

}

**private static void reverseColumn(int[][] matrix, int col)** {

int start = 0;

int end = matrix.length - 1;

while (start < end) {

// Swap the elements in the column

int temp = matrix[start][col];

matrix[start][col] = matrix[end][col];

matrix[end][col] = temp;

start++;

end--;

}

}

***SET B:***

**public class MatrixRotation** {

**public static void rotate(int[][] matrix)** {

int n = matrix.length;

// Transpose

for (int i = 0; i < n; i++) {

for (int j = i; j < n; j++) {

// Swap element at index (i, j) with element at index (j, i)

int temp = matrix[i][j];

matrix[i][j] = matrix[j][i];

matrix[j][i] = temp;

}

}

// Reverse

for (int i = 0; i < n; i++) {

reverseRow(matrix, i);

}

}

**private static void reverseRow(int[][] matrix, int row)** {

int start = 0;

int end = matrix.length - 1;

while (start < end) {

// Swap elements in the row

int temp = matrix[row][start];

matrix[row][start] = matrix[row][end];

matrix[row][end] = temp;

start++;

end--;

}

}

**Rubric:**

| Correct traversal and loop iteration | 1 |
| --- | --- |
| Transpose correctly | 2.5 |
| Reverse correctly | 2.5 |
| In-place implementation | 3 |
| Handles edge cases (e.g., small matrix sizes like 1x1, 2x2) | 2 |