Q-1)

```
import java.util.ArrayList;
import java.util.List;

class Solution {
    public List<Integer> spiralOrder(int[][] matrix) {
        List<Integer> result = new ArrayList<>();

        if (matrix == null || matrix.length == 0 || matrix[0].length
== 0) {
            return result;
        }

        int rowStart = 0, rowEnd = matrix.length - 1;
        int colStart = 0, colEnd = matrix[0].length - 1;

        while (rowStart <= rowEnd && colStart <= colEnd) {
            for (int i = colStart; i <= colEnd; i++) {
                result.add(matrix[rowStart][i]);
            }
            rowStart++;

            for (int i = rowStart; i <= rowEnd; i++) {
                result.add(matrix[i][colEnd]);
            }
            colEnd--;

            if (rowStart <= rowEnd) {
                for (int i = colEnd; i >= colStart; i--) {
                    result.add(matrix[rowEnd][i]);
                }
                rowEnd--;
            }

            if (colStart <= colEnd) {
                for (int i = rowEnd; i >= rowStart; i--) {
                    result.add(matrix[i][colStart]);
                }
                colStart++;
            }
        }

        return result;
    }
}

public class SpiralMatrix{
    }
```

Q-2)
```java
import java.util.*;

class Solution {
    public int[] kWeakestRows(int[][] mat, int k) {
        List<RowStrength> rowStrengths = new ArrayList<>();

        for (int i = 0; i < mat.length; i++) {
            int strength = countSoldiers(mat[i]);
            rowStrengths.add(new RowStrength(i, strength));
        }

        Collections.sort(rowStrengths);

        int[] result = new int[k];
        for (int i = 0; i < k; i++) {
            result[i] = rowStrengths.get(i).index;
        }

        return result;
    }

    private int countSoldiers(int[] row) {
        int count = 0;
        for (int num : row) {
            if (num == 1) {
                count++;
            } else {
                break; // Since soldiers appear before civilians
            }
        }
        return count;
    }

    class RowStrength implements Comparable<RowStrength> {
        int index;
        int strength;

        public RowStrength(int index, int strength) {
            this.index = index;
            this.strength = strength;
        }

        @Override
        public int compareTo(RowStrength other) {
            if (this.strength != other.strength) {
                return Integer.compare(this.strength,
other.strength);
            } else {
                    return Integer.compare(this.index, other.index);
                }
            }
        }
}
public class KWeakestRows{

}
```

Q-3)

```java
class Solution {
    public void setZeroes(int[][] matrix) {
        int m = matrix.length;
        int n = matrix[0].length;

        boolean firstRowZero = false;
        boolean firstColZero = false;

        // Step 1: Check if the first row contains zero
        for (int j = 0; j < n; j++) {
            if (matrix[0][j] == 0) {
                firstRowZero = true;
                break;
            }
        }

        // Step 2: Check if the first column contains zero
        for (int i = 0; i < m; i++) {
            if (matrix[i][0] == 0) {
                firstColZero = true;
                break;
            }
        }

        // Step 3: Mark rows and columns for zeroing
        for (int i = 1; i < m; i++) {
            for (int j = 1; j < n; j++) {
                if (matrix[i][j] == 0) {
                    matrix[0][j] = 0; // Mark the column
                    matrix[i][0] = 0; // Mark the row
                }
            }
        }

        // Step 4: Zero out marked rows (except the first row)
        for (int i = 1; i < m; i++) {
            if (matrix[i][0] == 0) {
                for (int j = 1; j < n; j++) {
                    matrix[i][j] = 0;
                }
            }
        }

        // Step 5: Zero out marked columns (except the first column)
        for (int j = 1; j < n; j++) {
            if (matrix[0][j] == 0) {
                for (int i = 1; i < m; i++) {
                    matrix[i][j] = 0;
                }
            }
        }

        // Step 6: Zero out the first row (if necessary)
        if (firstRowZero) {
            for (int j = 0; j < n; j++) {
                matrix[0][j] = 0;
            }
        }

        // Step 7: Zero out the first column (if necessary)
        if (firstColZero) {
            for (int i = 0; i < m; i++) {
                matrix[i][0] = 0;
            }
        }
    }
}
public class SetMatrixZeroes{

}
```

Q-4)

```java
class Solution {
    public void matrixAddition(int[][] matrixA, int[][] matrixB) {
        int n = matrixA.length;
        int[][] result = new int[n][n];

        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                result[i][j] = matrixA[i][j] + matrixB[i][j];
            }
        }
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                System.out.print(result[i][j] + " ");
            }
            System.out.println();
        }
    }
}

public class Addtwosquarematrices{
    public static void main(String[] args) {

    }
}
```

Q-5)

```java
class Solution {
    public int diagonalSum(int[][] mat) {
        int n = mat.length;
        int sum = 0;

        for (int i = 0; i < n; i++) {
            sum += mat[i][i];
            sum += mat[i][n - 1 - i];
        }
        if (n % 2 == 1) {
            int center = n / 2;
            sum -= mat[center][center];
        }

        return sum;
    }
}

public class MatrixDiagonalSum{
    public static void main(String[] args) {

    }
}
```

Q-6)

```
class Solution {
    public int[][] matrixMultiplication(int[][] matrixA, int[][]
matrixB) {
        int n = matrixA.length;
        int[][] result = new int[n][n];

        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                int sum = 0;
                for (int k = 0; k < n; k++) {
                    sum += matrixA[i][k] * matrixB[k][j];
                }
                result[i][j] = sum;
            }
        }

        return result;
    }
}

public class Multiply2matrices{
    public static void main(String[] args) {

    }
}
```