public class BoardProblemSolver {

private static final int BOARD\_SIZE = 4;

private static final int EMPTY = 0;

private static final int QUEEN = 1;

public static void main(String[] args) {

int[][] board = new int[BOARD\_SIZE][BOARD\_SIZE];

if (solveBoardProblem(board, 0)) {

System.out.println("Solution found:");

printBoard(board);

} else {

System.out.println("No solution found.");

}

}

public static boolean solveBoardProblem(int[][] board, int col) {

if (col >= BOARD\_SIZE) {

return true; // All queens have been placed

}

for (int row = 0; row < BOARD\_SIZE; row++) {

if (isSafe(board, row, col)) {

board[row][col] = QUEEN;

if (solveBoardProblem(board, col + 1)) {

return true;

}

// Backtrack if the current arrangement is not valid

board[row][col] = EMPTY;

}

}

return false; // No valid arrangement found

}

public static boolean isSafe(int[][] board, int row, int col) {

// Check row on the left side

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

if (board[row][i] == QUEEN) {

return false;

}

}

// Check upper diagonal on the left side

for (int i = row, j = col; i >= 0 && j >= 0; i--, j--) {

if (board[i][j] == QUEEN) {

return false;

}

}

// Check lower diagonal on the left side

for (int i = row, j = col; i < BOARD\_SIZE && j >= 0; i++, j--) {

if (board[i][j] == QUEEN) {

return false;

}

}

return true; // No conflicts found

}

public static void printBoard(int[][] board) {

for (int i = 0; i < BOARD\_SIZE; i++) {

for (int j = 0; j < BOARD\_SIZE; j++) {

System.out.print(board[i][j] + " ");

}

System.out.println();

}

}

}