ASSIGNMENT NO. : 5

public class NQueens {

private int N; // Number of queens

private int[][] board;

public NQueens(int n) {

N = n;

board = new int[N][N];}

private void printBoard() {

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

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

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

System.out.println();}

System.out.println()

// Function to check if a queen can be placed at board[row][col]

private boolean isSafe(int row, int col) {

// Check this column on upper side

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

if (board[i][col] == 1) {

return false;}}

// Check upper diagonal on left side

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

if (board[i][j] == 1) {

return false } }

// Check upper diagonal on right side

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

if (board[i][j] == 1) {

return false;} }

return true;}

// Backtracking function to solve the N-Queens problem

private boolean solveNQueens(int row) {

if (row >= N) {

return true; // All queens are placed}

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

if (isSafe(row, col)) {

board[row][col] = 1; // Place queen

// Recur to place the rest of the queens

if (solveNQueens(row + 1)) {

return true;}

// If placing queen in this position doesn't lead to a solution,

// backtrack

board[row][col] = 0;}}

return false; // No valid position found}

// Function to solve the N-Queens problem starting from a fixed position for the first queen

public void solve(int firstRow, int firstCol) {

// Place the first queen

board[firstRow][firstCol] = 1;

if (!solveNQueens(1)) {

System.out.println("Solution does not exist.");

} else { printBoard();

public static void main(String[] args) {

int n = 4; // Change this value for larger or smaller boards

int firstRow = 0; // First queen's row

int firstCol = 1; // First queen's column

NQueens nQueens = new NQueens(n);

nQueens.solve(firstRow, firstCol);

}}

0 1 0 0

0 0 0 1

1 0 0 0

0 0 1 0