**Name: Surwade Trisharan Rajesh**

**Roll no.: 48**

// Write a program to find all solutions for 8-queen problem using backtracking.

#include <iostream>

#include <vector>

using namespace std;

bool isSafe(vector<vector<int>>& board, int row, int col, int N) {

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

if (board[row][i])

return false;

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

if (board[i][j])

return false;

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

if (board[i][j])

return false;

return true;

}

bool solveNQueens(vector<vector<int>>& board, int col, int N, vector<vector<vector<int>>>& solutions) {

if (col == N) {

solutions.push\_back(board);

return true;

}

bool res = false;

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

if (isSafe(board, i, col, N)) {

board[i][col] = 1;

res = solveNQueens(board, col + 1, N, solutions) || res;

board[i][col] = 0;

}

}

return res;

}

void printSolution(vector<vector<int>>& board) {

int N = board.size();

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

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

cout << board[i][j] << " ";

}

cout << endl;

}

cout << endl;

}

int main() {

int N = 8;

vector<vector<int>> board(N, vector<int>(N, 0));

vector<vector<vector<int>>> solutions;

solveNQueens(board, 0, N, solutions);

int numSolutions = solutions.size();

cout << "Total solutions: " << numSolutions << endl;

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

cout << "Solution " << i + 1 << ":\n";

printSolution(solutions[i]);

}

return 0;

}