//Om Jadhav

// N-queue

#include <iostream>

#include <vector>

#include <cmath>

using namespace std;

bool isSafe(int board[][10], int row, int col, int n) {

// check if there is a queen in the same row

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

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

return false;

}

}

// check if there is a queen in the same column

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

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

return false;

}

}

// check if there is a queen on the diagonal

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

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

return false;

}

}

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

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

return false;

}

}

// it's safe to place a queen at (row, col)

return true;

}

bool backtrack(int board[][10], int row, int n) {

if (row == n) {

return true;

}

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

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

board[row][col] = 1;

if (backtrack(board, row+1, n)) {

return true;

}

board[row][col] = 0; // backtrack

}

}

return false;

}

int main() {

int n;

cout << "Enter the size of the chessboard: ";

cin >> n;

int board[10][10] = {0};

if (backtrack(board, 0, n)) {

// print the first solution found

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

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

if(board[i][j] == 0){

cout << "\_ ";

}

else{

cout << "Q ";

}

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

}

cout << endl;

}

} else {

cout << "No solution found." << endl;

}

return 0;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* **OUTPUT** \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Enter the size of the chessboard: 10

Q \_ \_ \_ \_ \_ \_ \_ \_ \_

\_ \_ Q \_ \_ \_ \_ \_ \_ \_

\_ \_ \_ \_ \_ Q \_ \_ \_ \_

\_ \_ \_ \_ \_ \_ \_ Q \_ \_

\_ \_ \_ \_ \_ \_ \_ \_ \_ Q

\_ \_ \_ \_ Q \_ \_ \_ \_ \_

\_ \_ \_ \_ \_ \_ \_ \_ Q \_

\_ Q \_ \_ \_ \_ \_ \_ \_ \_

\_ \_ \_ Q \_ \_ \_ \_ \_ \_

\_ \_ \_ \_ \_ \_ Q \_ \_ \_

=== Code Execution Successful ===