Aim: write a program to implement n queens problem using backtracking

Algorithm:

1. Initialize a 2D array chess\_board[N][N] to represent the chess board, where N is the size of the chess board.
2. Define a function isSafe(chess\_board, row, col) to check if it is safe to place a queen in a given cell (row, col).
3. Define a function solveNQUtil(chess\_board, col) which uses backtracking to place queens in each column.
4. If all queens are placed successfully, the function returns true.
5. If a queen cannot be placed in the current column, the function returns false.
6. In the main function, take the input n from the user and call solveNQUtil(chess\_board, 0).
7. If solveNQUtil returns false, the solution does not exist.
8. If solveNQUtil returns true, the solution exists and print the chess board.

Note: The chess\_board is initialized to all zeros and the cells with 1 represent the placement of a queen.

Source Code:

#include<stdio.h>

#include<stdbool.h>

*int* N;

*int* chess\_board[100][100];

*int* count = 0;

*void* printSolution(*int* chess\_board[100][100])

{

    printf("Solution %d:\n", ++count);

    for (*int* i = 0; i < N; i++)

    {

        for (*int* j = 0; j < N; j++)

            printf(" %d ", chess\_board[i][j]);

        printf("\n");

    }

}

*bool* isSafe(*int* chess\_board[100][100], *int* row, *int* col)

{

*int* i, j;

    for (i = 0; i < col; i++)

        if (chess\_board[row][i])

            return *false*;

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

        if (chess\_board[i][j])

            return *false*;

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

        if (chess\_board[i][j])

            return *false*;

    return *true*;

}

*bool* solveNQUtil(*int* chess\_board[100][100], *int* col)

{

    if (col == N)

    {

        printSolution(chess\_board);

        return *false*;

    }

    for (*int* i = 0; i < N; i++)

    {

        if ( isSafe(chess\_board, i, col) )

        {

            chess\_board[i][col] = 1;

            if ( solveNQUtil(chess\_board, col + 1) == *true* )

                return *true*;

            chess\_board[i][col] = 0;

        }

    }

    return *false*;

}

*int* main()

{

    printf("Enter the value of n for n-queen problem: ");

    scanf("%d", &N);

    solveNQUtil(chess\_board, 0);

    return 0;

}

Output:

Enter the value of n for n-queen problem: 4

Solution 1:

0 0 1 0

1 0 0 0

0 0 0 1

0 1 0 0

Solution 2:

0 1 0 0

0 0 0 1

1 0 0 0

0 0 1 0