5. N-queen

Time Complexity: O(N!)

Algorithm

1. Input the size of the chessboard (`N`).
2. Initialize a chessboard of size `NxN`.
3. Start with the first row and place a queen in the first column.
4. Move to the next row and place the queen in the next available column that doesn't attack any other queen.
5. Repeat step 4 recursively for each row.
6. If all queens are placed without conflict, return true; otherwise, backtrack and try another column for the current row.
7. Continue backtracking until all possible solutions are found.

Code:

```
#include <stdio.h>
#include <stdbool.h>

#define N 10
bool isSafe(int board[N][N], int row, int col)
{
    int i, j;

    // Check if there's a queen in the same row
    for (i = 0; i < col; i++)
        if (board[row][i])
            return false;

    // Check upper left diagonal
    for (i = row, j = col; i >= 0 && j >= 0; i--, j--)
        if (board[i][j])
            return false;

    // Check lower left diagonal
    for (i = row, j = col; j >= 0 && i < N; i++, j--)
        if (board[i][j])
            return false;
```

```
    return true;
}

bool solveNQueensUtil(int board[N][N], int col)
{
    // If all queens are placed, return true
    if (col >= N)
        return true;

    // Try placing this queen in all rows one by one
    for (int i = 0; i < N; i++)
            {
        if (isSafe(board, i, col))
                    {
            // Place the queen
            board[i][col] = 1;
            // Recur to place the rest of the queens
            if (solveNQueensUtil(board, col + 1))
                return true;
            // If placing queen in board[i][col] doesn't lead to a solution, then remove queen from
board[i][col]
            board[i][col] = 0; // backtrack
        }
    }
    // If the queen can't be placed in any row in this column, return false
    return false;
}

void printSolution(int board[N][N])
{
    for (int i = 0; i < N; i++)
            {
        for (int j = 0; j < N; j++)
                    {
            if (board[i][j])
                printf("Q ");
            else
                printf("_ ");
        }
        printf("\n");
    }
}
```

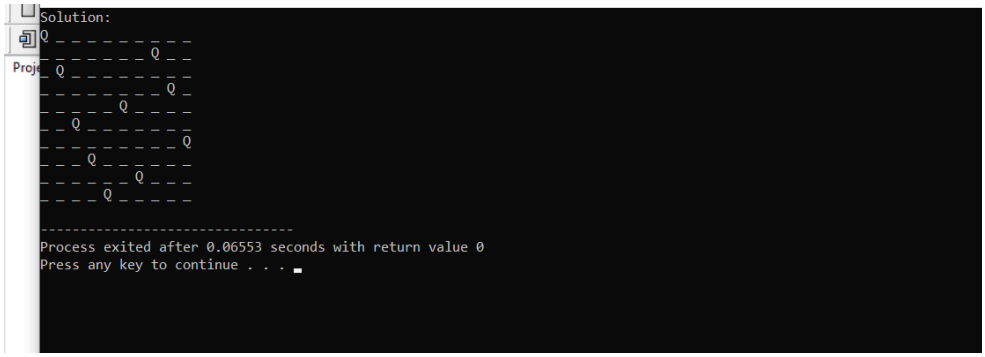
```
void solveNQueens()
{
    int board[N][N] = {0};

    if (!solveNQueensUtil(board, 0))
            {
        printf("Solution does not exist\n");
        return;
    }

    printf("Solution:\n");
    printSolution(board);
}

int main()
{
    solveNQueens();
    return 0;
}
```

input/output:



Expl: