N-Queen’s Problem

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

#define N 4  // You can change this to any value of N

// Function to print the solution (optional, can be removed if not needed)
void printSolution(int board[N][N]) {
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            if (board[i][j] == 1)
                printf(" Q ");
            else
                printf(" . ");
        }
        printf("\n");
    }
    printf("\n");
}

// Function to check if a queen can be placed on board[row][col]
bool isSafe(int board[N][N], int row, int col) {
    int i, j;

    // Check this row on left side
    for (i = 0; i < col; i++)
```

```
        if (board[row][i])
            return false;

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

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

    return true;
}

// Function to solve the N-Queens problem using backtracking and count solutions
int solveNQUtil(int board[N][N], int col) {
    if (col >= N)
        return 1; // All queens placed successfully

    int count = 0;

    for (int i = 0; i < N; i++) {
        if (isSafe(board, i, col)) {
```

```
            board[i][col] = 1;

            count += solveNQUtil(board, col + 1);

            board[i][col] = 0; // Backtrack
        }
    }

    return count;
}

// Function to count solutions for the N-Queens problem
int countNQSolutions() {
    int board[N][N] = {0};
    return solveNQUtil(board, 0);
}

int main() {
    int numSolutions = countNQSolutions();
    printf("Number of solutions for %d-Queens problem: %d\n", N,
numSolutions);
    return 0;
}
```

Output:

Number of solutions for 4-Queens problem: 2