**Practical-14**

**AIM:** To implement N Queen using Backtracking in C language.

**SOFTWARE REQUIRED:** Vs Code

**PSEUDO CODE:**

function isSafe(board, row, col, n):

for i from 0 to col - 1:

if board[row][i] is 1:

return false # Check the same row

for i from row, j from col to 0, while i >= 0 and j >= 0:

if board[i][j] is 1:

return false # Check upper left diagonal

for i from row, j from col to n, while i < n and j >= 0:

if board[i][j] is 1:

return false # Check lower left diagonal

return true

function solveNQueens(board, col, n):

if col >= n:

return true # All queens are placed

for row from 0 to n - 1:

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

board[row][col] = 1 # Place queen at [row][col]

if solveNQueens(board, col + 1, n):

return true # If a solution is found, return true

board[row][col] = 0 # Backtrack if no solution found

return false # No solution found for this configuration

function printBoard(board, n):

for i from 0 to n - 1:

for j from 0 to n - 1:

print(board[i][j], " ")

print("\n")

function main():

n = input("Enter the number of queens (N):")

initialize an empty chessboard "board" of size n x n

if solveNQueens(board, 0, n):

print("Solution found:")

printBoard(board, n)

else:

print("No solution found for N =", n)

main()

**CODE:**

#include <stdio.h>

#include <stdbool.h>

#define N 10

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

    int i, j;

    // Check the row on the left side

    for (i = 0; i < col; i++) {

        if (board[row][i])

            return false;

    }

    // Check upper diagonal on the left side

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

        if (board[i][j])

            return false;

    }

    // Check lower diagonal on the left side

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

        if (board[i][j])

            return false;

    }

    return true;

}

bool solveNQueens(int board[N][N], int col, int n) {

    if (col >= n)

        return true;

    for (int i = 0; i < n; i++) {

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

            board[i][col] = 1;

            if (solveNQueens(board, col + 1, n))

                return true;

            board[i][col] = 0;

        }

    }

    return false;

}

void printBoard(int board[N][N], int n) {

    for (int i = 0; i < n; i++) {

        for (int j = 0; j < n; j++) {

            printf("%d ", board[i][j]);

        }

        printf("\n");

    }

}

int main() {

    int n;

    printf("Ananta Walli, A2305221322");

    printf("\nEnter the number of queens: ");

    scanf("%d", &n);

    int board[N][N] = {0};

    if (solveNQueens(board, 0, n)) {

        printf("The solution will be:\n");

        printBoard(board, n);

    } else {

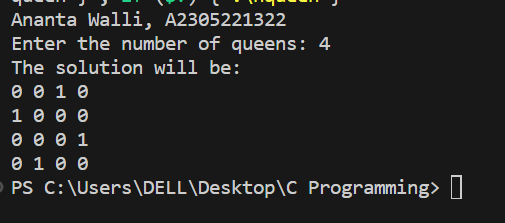
        printf("Oh no! no solution found for N = %d.\n", n);

    }

    return 0;

}

**OUTPUT:**

****

**RESULT:** The above code implements N Queen using Backtracking in C programming.