1. **N-Queens Problem**

**Code:**

#include <stdio.h>

#include <stdbool.h>

// Function to print the solution

void printSolution(int board[][10], int N) {

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

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

printf("%2d ", board[i][j]);

}

printf("\n");

}

}

// Function to check if a queen can be placed on the board at position (row, col)

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

int i, j;

// Check this 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; j >= 0 && i < N; i++, j--) {

if (board[i][j])

return false;

}

return true;

}

// Recursive function to solve N-Queens problem using backtracking

bool solveNQueensUtil(int board[][10], int col, int N) {

// If all queens are placed, return true

if (col >= N)

return true;

// Consider this column and try placing this queen in all rows one by one

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

// Check if the queen can be placed on the board[i][col]

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

// Place this queen in board[i][col]

board[i][col] = 1;

// Recur to place rest of the queens

if (solveNQueensUtil(board, col + 1, N))

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 cannot be placed in any row in this column, then return false

return false;

}

// Function to solve the N-Queens problem

bool solveNQueens(int N) {

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

if (solveNQueensUtil(board, 0, N) == false) {

printf("Solution does not exist");

return false;

}

printSolution(board, N);

return true;

}

// Main function

int main() {

int N;

printf("Enter the size of the board: ");

scanf("%d", &N);

if (N <= 0 || N > 10) {

printf("Invalid input. Size should be between 1 and 10.\n");

return 1;

}

solveNQueens(N);

return 0;

}

**Output:**

Enter the size of the board: 8

1 0 0 0 0 0 0 0

0 0 0 0 0 0 1 0

0 0 0 0 1 0 0 0

0 0 0 0 0 0 0 1

0 1 0 0 0 0 0 0

0 0 0 1 0 0 0 0

0 0 0 0 0 1 0 0

0 0 1 0 0 0 0 0

--------------------------------

Process exited after 1.396 seconds with return value 0

Press any key to continue . . .

