**Practical:10**

* Implement N Queen's problem using Backtracking.

**Code:**

#include <stdio.h>

#include <stdbool.h>

#define N 8

bool is\_safe(int board[N][N], int row, int col)

{

    int i, j;

    for (i = 0; i < col; i++) {

        if (board[row][i]) {

            return false;

        }

    }

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

        if (board[i][j]) {

            return false;

        }

    }

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

        if (board[i][j]) {

            return false;

        }

    }

    return true;

}

bool solve\_n\_queens(int board[N][N], int col)

{

    if (col >= N) {

        return true;

    }

    int i;

    for (i = 0; i < N; i++) {

        if (is\_safe(board, i, col)) {

            board[i][col] = 1;

            if (solve\_n\_queens(board, col + 1)) {

                return true;

            }

            board[i][col] = 0;

        }

    }

    return false;

}

void print\_board(int board[N][N])

{

    int i, j;

    for (i = 0; i < N; i++) {

        for (j = 0; j < N; j++) {

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

        }

        printf("\n");

    }

}

int main()

{

    int board[N][N] = {0};

    if (solve\_n\_queens(board, 0)) {

        printf("Solution:\n");

        print\_board(board);

    } else {

        printf("No solution exists for a %dx%d board.\n", N, N);

    }

    return 0;

}

**Output:**

