Practical:10

- Implement N Queen's problem using Backtracking.

Code:

```c
#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:

```
Solution:
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
```