**9913 Mark Lopes**

**SE Comps A Batch-C**

**N-Queen:-**

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

int n;

int col[20] = {0}; // Array to store column positions of queens

int count = 0;

// Function to print the solution (board configuration)

void solution()

{

    int i, j;

    for (i = 1; i <= n; i++)

    {

        for (j = 1; j < col[i]; j++)

            printf("x\t"); // Print 'x' for empty spaces

        printf("q%d\t", i); // Print 'q' for queens

        for (; j <= n; j++)

        {

            printf("x\t"); // Print 'x' for empty spaces

        }

        printf("\n");

    }

}

// Function to check if placing a queen at (q, c) is safe

int place(int q, int c)

{

    int k;

    for (k = 1; k <= q - 1; k++)

    {

        if (col[k] == c || abs(k - q) == abs(col[k] - c))

        {

            return 0;

        }

    }

    return 1;

}

// Function to place queens on the board (backtracking approach)

void nqueen(int q)

{

    int c;

    for (c = 1; c <= n; c++)

    {

        if (place(q, c))

        {

            col[q] = c; // Place queen at row q, column c

            if (q == n)

            {

                count++;

                printf("\nSolution number: %d\n", count);

                solution(); // Print the solution

            }

            else

            {

                nqueen(q + 1); // Recursively place the next queen

            }

        }

    }

}

int main()

{

    printf("Enter the number of queens: ");

    scanf("%d", &n);

    if (n <= 0)

    {

        printf("Invalid number of queens. Please enter a positive integer.\n");

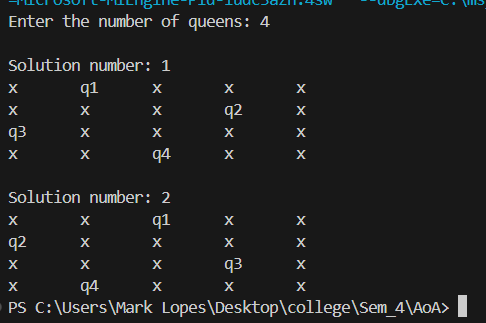
        return 1;

    }

    nqueen(1); // Start placing queens from the first row

    return 0;

}

****