**Job No: 04**

**Job Name: Write a program for N-Queen Problem**

**Theory:** The N-Queens problem is a classic puzzle that involves placing N queens on an N x N chessboard, such that no two queens attack each other.

A queen can attack horizontally, vertically, and diagonally. To solve the N-Queens problem, we typically use backtracking.

**Code:**

class **NQueensSolver** {

  constructor(n) {

    this.n = n;

    this.board = [];

    for (let i = 0; i < n; i++) {

      this.board[i] = [];

      for (let j = 0; j < n; j++) {

        this.board[i][j] = ".";

      }

    }

    this.solutions = [];

  }

**solve**() {

    this.**backtrack**(0);

    return this.solutions;

  }

**isSafe**(row, col) {

    for (let i = 0; i < row; i++) {

      if (this.board[i][col] === "Q") {

        return false;

      }

    }

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

      if (this.board[i][j] === "Q") {

        return false;

      }

    }

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

      if (this.board[i][j] === "Q") {

        return false;

      }

    }

    return true;

  }

**backtrack**(row) {

    if (row === this.n) {

      const solution = [];

      for (let i = 0; i < this.n; i++) {

        solution.**push**(this.board[i].**join**(""));

      }

      this.solutions.**push**(solution);

      return;

}

    for (let col = 0; col < this.n; col++) {

      if (this.**isSafe**(row, col)) {

        this.board[row][col] = "Q";

        this.**backtrack**(row + 1);

        this.board[row][col] = ".";

      }

    }

  }

}

const n = +**prompt**("Enter the value of n:");

const solver = new **NQueensSolver**(n);

const solutions = solver.**solve**();

console.**log**(`Found ${solutions.length} solutions for ${n}-queens problem:`);

solutions.**forEach**((solution, index) => {

  console.**log**(`Solution ${index + 1}:`);

  solution.**forEach**((row) => {

    console.**log**(row);

  });

  console.**log**();

});

**Input/Output:**

