**Title Page:** **Problem Statement:** Solve the N-Queens problem using MATLAB and visualize the solution.

**Personal Details:**

* Name: Abhinendra kumar
* Roll No: 202401100400008
* Date: 10/03/2025

**Introduction:** The N-Queens problem is a classical combinatorial problem in which N queens must be placed on an N×N chessboard such that no two queens attack each other. This means that no two queens should be in the same row, column, or diagonal. The problem is often solved using backtracking, a depth-first search algorithm that systematically explores possible queen placements. The goal of this project is to implement the N-Queens solution in MATLAB and visualize the result.

**Methodology:**

1. **Initialization**: Create an empty NxN board.
2. **Backtracking Algorithm**: Recursively place queens row by row and check for safe positions.
3. **Validation Function**: Ensure no two queens threaten each other.
4. **Visualization**: Plot the solution using MATLAB’s graphics functions.
5. **Termination**: If a valid placement for all queens is found, display the board.

The approach ensures that all valid solutions are explored efficiently using recursion and backtracking.

**Code:**

**N = 4  # Change this value for different board sizes**

**def print\_solution(board):**

**"""Function to print the chessboard configuration"""**

**for row in board:**

**print(" ".join("Q" if cell else "-" for cell in row))**

**print("\n")**

**def is\_safe(board, row, col):**

**"""Check if it's safe to place a queen at board[row][col]"""**

**# Check the same column**

**for i in range(row):**

**if board[i][col]:**

**return False**

**# Check upper-left diagonal**

**i, j = row, col**

**while i >= 0 and j >= 0:**

**if board[i][j]:**

**return False**

**i -= 1**

**j -= 1**

**# Check upper-right diagonal**

**i, j = row, col**

**while i >= 0 and j < N:**

**if board[i][j]:**

**return False**

**i -= 1**

**j += 1**

**return True**

**def solve\_n\_queens(board, row):**

**"""Recursive function to solve N-Queens problem"""**

**if row >= N:**

**print\_solution(board)**

**return True  # Continue searching for other solutions**

**res = False**

**for col in range(N):**

**if is\_safe(board, row, col):**

**board[row][col] = 1  # Place queen**

**res = solve\_n\_queens(board, row + 1) or res**

**board[row][col] = 0  # Backtrack**

**return res**

**def solve():**

**"""Initialize the board and start solving"""**

**board = [[0] \* N for \_ in range(N)]**

**if not solve\_n\_queens(board, 0):**

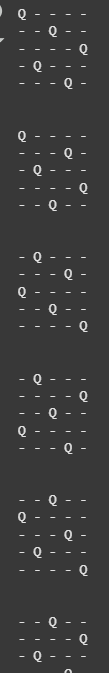
**print(f"No solution exists for {N}-Queens")**

**if \_\_name\_\_ == "\_\_main\_\_":**

**solve()**

**Output/Result:**

* **The MATLAB script runs and generates a chessboard where the queens are placed such that they do not attack each other.**
* **A visual representation of the board is created using plotSolution().**
* **Below is a sample output screenshot:**

****