**N-Queens Visualizer Report**

**1. Introduction**

The N-Queens problem is a classic puzzle in the field of computer science and mathematics. It involves placing N chess queens on an N×N chessboard so that no two queens threaten each other. This means that no two queens can share the same row, column, or diagonal. The problem is a specific case of the more general m-Queens problem and has applications in various fields, including algorithm design, artificial intelligence, and operations research.

The purpose of the N-Queens visualizer is to provide a clear and interactive way to understand how the solution to this problem can be found using backtracking algorithms. By visualizing the steps, users can gain deeper insights into the problem-solving process, see how the algorithm proceeds step-by-step, and appreciate the complexity of the solution.

**2. Background**

The N-Queens problem has a rich history, originating in 1848 when it was first proposed by Max Bezzel. Over the years, numerous methods have been developed to solve the problem, ranging from brute-force search to more sophisticated backtracking algorithms.

Mathematically, the problem can be defined as placing N queens on an N×N chessboard so that no two queens attack each other. The challenge lies in the combinatorial explosion of possibilities, as the number of ways to place N queens grows factorially with N.

**3. Design and Implementation**

The provided code implements an N-Queens visualizer using C++. The program uses the following components:

* **Programming Language**: C++ for its efficiency and control over system resources.
* **User Interface**: The console is used to print the board and visualize the steps.
* **Key Features**: The program visualizes each step of the algorithm, including the placement of queens, backtracking steps, and final solutions.

Here’s a breakdown of the key functions:

1. **printBoard**: This function prints the current state of the board, indicating the positions of the queens.
2. **isSafe**: This function checks if placing a queen at a specific position is safe, i.e., it does not result in any conflicts.
3. **solveNQueensUtil**: This is the recursive function that uses backtracking to find all solutions to the N-Queens problem.
4. **solveNQueens**: This function initializes the board and starts the recursive process.