**A purple circle with dots and lines

Description automatically generated**

**NAME:**

**Basit Abbas**

**ROLL NUMBER:**

**SU92-BSDSM-F23- 013**

**PROGRAM:**

**BS DATA SCIENCE**

**SUMBITTED TO:**

**SIR RASIKH**

***TASK : 04***

**N-Queens Problem Implementation Report**

**1. Introduction** The N-Queens problem is a classic combinatorial optimization problem that involves placing N queens on an N×N chessboard such that no two queens attack each other. This implementation uses a backtracking approach to find a valid arrangement of queens.

**2. Approach** The algorithm places queens row by row, ensuring that each placed queen does not attack another. It uses recursive backtracking to explore possible placements and backtracks when a conflict is encountered.

**3. Code Explanation**

* **State Representation:** The chessboard is represented as an N×N matrix where 1 denotes a queen and 0 represents an empty space.
* **Validation Functions:**
  + check\_col(board, row, column): Ensures that no queen is placed in the same column above the current row.
  + check\_daig(board, row, column): Ensures that no queen is placed diagonally in both left and right directions.
* **Recursive Backtracking Function:**
  + nqn(board, row): Attempts to place a queen in each column of the given row. If a valid placement is found, it proceeds to the next row recursively. If no placement is possible, it backtracks.
* **Base Case:** When all rows are filled successfully, the function returns True and prints the board configuration.

**4. Implementation Issues and Fixes**

* The diagonal checking logic was initially flawed, and proper index handling was corrected.
* The board initialization was verified to prevent unintended modifications.
* The function return values were optimized to ensure the algorithm stops upon finding a valid solution.

**5. Conclusion** This implementation successfully solves the N-Queens problem using an efficient backtracking approach. The use of column and diagonal checks ensures that all placed queens follow the constraints, leading to an optimized solution.