Assignment 3

**Explanation of the Backtracking Approach:**

* **is\_safe(board, row, col)**: This function checks if it's safe to place a queen at a given position **(row, col)** on the chessboard. It checks the column, upper-left diagonal, and upper-right diagonal for any other queens.
* **solve(row)**: This recursive function tries to place queens row by row. If it successfully places all queens, it adds the current board state to the solutions list.
* The **board** represents the chessboard as a 2D list with 'Q' for queens and '.' for empty squares.
* We iterate through columns for each row and call **is\_safe** to check if a queen can be placed at that position. If yes, we place the queen ('Q') and continue solving for the next row. If not, we backtrack by changing the placement back to an empty square ('.').
* **print\_solutions** prints all found solutions in a human-readable format.

**Time Complexity Analysis:**

The time complexity of this backtracking algorithm is exponential, specifically O(N!). This is because for each row, we try all possible columns, leading to a large number of recursive calls