A8_Assignment

Name: Tanaya Bhore

Class: TE A Batch: A

Roll No.: 3101016

```
class NQueens:
    def __init__(self, N):
        self.N = N
        self.board = [[0] * N for _ in range(N)]
        self.column = [False] * N
        self.diagonal1 = [False] * (2 * N - 1)
        self.diagonal2 = [False] * (2 * N - 1)

    def solve(self, row=0):
        if row == self.N:
            self.print_solution()
            return True

        for col in range(self.N):
            if not self.is_safe(row, col):
                continue

            self.place_queen(row, col)
            if self.solve(row + 1):
                return True  # Return true to get one solution only
            self.remove_queen(row, col)

        return False

    def is_safe(self, row, col):
        return not (self.column[col] or self.diagonal1[row - col + self.N - 1] or self.diagonal2[row + col])
```

```
    def place_queen(self, row, col):
        self.board[row][col] = 1
        self.column[col] = True
        self.diagonal1[row - col + self.N - 1] = True
        self.diagonal2[row + col] = True

    def remove_queen(self, row, col):
        self.board[row][col] = 0
        self.column[col] = False
        self.diagonal1[row - col + self.N - 1] = False
        self.diagonal2[row + col] = False

    def print_solution(self):
        for row in self.board:
            print(" ".join("Q" if col else "." for col in row))
        print("\n")

if __name__ == "__main__":
    N = 8  # You can change this value
    solver = NQueens(N)
    if not solver.solve():
        print("No solution exists.")
```