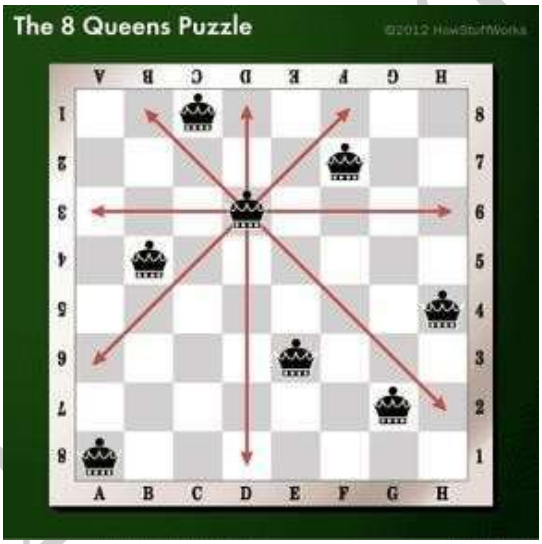
EX.NO: **DATE:**

8- QUEENS PROBLEM

AIM :
To implement an 8-Queens problem using Python.

You are given an 8x8 board; find a way to place 8 queens such that no queen can attack any other queen on the chessboard. A queen can only be attacked if it lies on the same row, same column, or the same diagonal as any other queen. Print all the possible configurations.
To solve this problem, we will make use of the Backtracking algorithm. The backtracking algorithm, in general checks all possible configurations and test whether the required result is obtained or not. For the given problem, we will explore all possible positions the queens can be relatively placed at. The solution will be correct when the number of placed queens = 8.



CODE:

```
def is_safe(board, row, col):
    for i in range(row):
        if board[i] == col or \
           board[i] - i == col - row or \
           board[i] + i == col + row:
            return False
    return True

def solve_queens(board, row, solutions):
    if row == 8:
        solutions.append(board.copy())
        return

    for col in range(8):
        if is_safe(board, row, col):
            board[row] = col
            solve_queens(board, row + 1, solutions)
            board[row] = -1

def print_solutions(solutions):
    for solution in solutions:
        for row in range(8):
            board_row = ['Q' if col == solution[row] else '.' for col in range(8)]
            print(' '.join(board_row))
        print()

def eight_queens():
    solutions = []
    board = [-1] * 8

    solve_queens(board, 0, solutions)

    print_solutions(solutions)

eight_queens()
```

OUTPUT:

```
Enter the number of queens
8
[1, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 1, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 1]
[0, 0, 0, 0, 0, 1, 0, 0]
[0, 0, 1, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 1, 0]
[0, 1, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 1, 0, 0, 0, 0]
```


RESULT:

Thus, the code has been successfully executed, and the output has been verified successfully.