EX.NO: **DATE:**

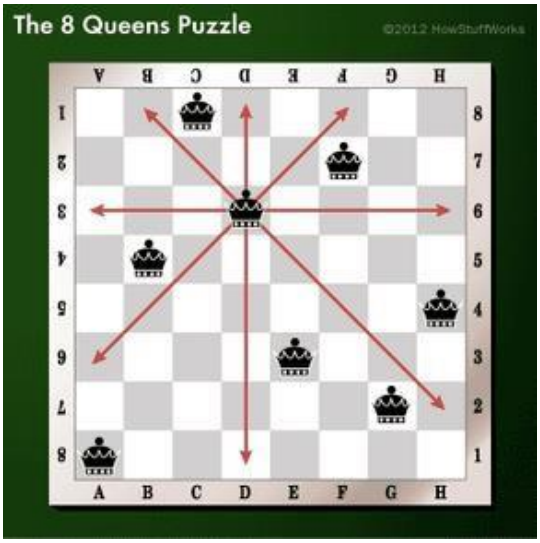
8- QUEENS PROBLEM

AIM :

To implement an 8-Queesns 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:

```python
def is_safe(board, row, col, N):
    # Check this row on the left side
    for i in range(col):
        if board[row][i] == 1:
            return False

    # Check upper diagonal on the left side
    for i, j in zip(range(row, -1, -1), range(col, -1, -1)):
        if board[i][j] == 1:
            return False

    # Check lower diagonal on the left side
    for i, j in zip(range(row, N, 1), range(col, -1, -1)):
        if board[i][j] == 1:
            return False

    return True

def solve_n_queens(board, col, N):
    # Base case: If all queens are placed
    if col >= N:
        return True

    # Consider this column and try placing the queen in all rows one by one
    for i in range(N):
        if is_safe(board, i, col, N):
            # Place this queen in board[i][col]
            board[i][col] = 1

            # Recur to place the rest of the queens
            if solve_n_queens(board, col + 1, N):
                return True

            # If placing queen in board[i][col] doesn't lead to a solution, then remove queen from board[i][col]
            board[i][col] = 0

    # If the queen cannot be placed in any row in this column col, then return false
    return False

def print_board(board, N):
    for row in board:
        print(" ".join(str(cell) for cell in row))

# Input from the user
N = int(input("Enter the size of the board (N): "))

# Initialize the board with all 0s
board = [[0] * N for _ in range(N)]

if solve_n_queens(board, 0, N):
    print("Solution found:")
    print_board(board, N)
else:
    print("No solution exists.")
```

OUTPUT:

```
Enter the size of the board (N): 8
Solution found:
1 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0
0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 1
0 1 0 0 0 0 0 0
0 0 0 1 0 0 0 0
0 0 0 0 0 1 0 0
0 0 1 0 0 0 0 0
```

RESULT:

Thus Program is Executed Successfully And Output is Verified.