207. Discuss the generalization of the N-Queens Problem to other board sizes and shapes, such as rectangular boards or boards with obstacles. Explain how the algorithm can be adapted to handle these variations and the additional constraints they introduce. Provide examples of solving generalized N-Queens Problems for different board configurations, such as an 8×10 board, a 5×5 board with obstacles, and a 6×6 board with restricted positions.

a. 8×10 Board:
   8 rows and 10 columns
   Output: Possible solution [1, 3, 5, 7, 9, 2, 4, 6]
   Explanation: Adapt the algorithm to place 8 queens on an 8×10 board, ensuring no two queens threaten each other.

PROGRAM:-

```
def is_safe(board, row, col):
  for i in range(col):
    if board[row][i] == 1:
      return False
  for i, j in zip(range(row, -1, -1), range(col, -1, -1)):
    if board[i][j] == 1:
      return False
  for i, j in zip(range(row, len(board), 1), range(col, -1, -1)):
    if board[i][j] == 1:
      return False
  return True

def solve_n_queens(board, col):
  if col >= len(board):
    return True
  for i in range(len(board)):
    if is_safe(board, i, col):
      board[i][col] = 1
      if solve_n_queens(board, col + 1):
        return True
      board[i][col] = 0
  return False

def print_solution(board):
  for row in board:
    print(row)

# 8x10 Board
board_8x10 = [[0 for _ in range(10)] for _ in range(8)]
if solve_n_queens(board_8x10, 0):
  print_solution(board_8x10)
else:
  print("No solution found.")
```

OUTPUT:-

```
[1, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 1, 0, 0, 0]
[0, 0, 0, 0, 1, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 1, 0, 0]
[0, 1, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 1, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 1, 0, 0, 0, 0]
[0, 0, 1, 0, 0, 0, 0, 0, 0, 0]

=== Code Execution Successful ===
```

TIME COMPLEXITY:-O(n!*n)