```
Def is_safe(board, row, col, n):
# چک کردن آیا می‌توان وزیری را در سلول (row, col) قرار داد یا خیر
# چک کردن ردیف افقی (سمت چپ)
    For i in range(col):
        If board[row][i] == 1:
            Return False

# چک کردن قطر بالا به چپ
    For i, j in zip(range(row, -1, -1), range(col, -1, -1)):
        If board[i][j] == 1:
            Return False

# چک کردن قطر پایین به چپ
    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_util(board, col, n):
# حالت پایه: اگر تمام وزیرها قرار گرفته باشند
    If col >= n:
        Return True

# برای هر سلول در ستون فعلی
    For i in range(n):
# چک کردن آیا می‌توان وزیر را در این سلول قرار داد
        If is_safe(board, i, col, n):
```

```
# قرار دادن وزیر در این سلول
        Board[i][col] = 1

# ادامه به جستجوی ستون بعدی
        If solve_n_queens_util(board, col + 1, n):
            Return True

# اگر قرار گرفتن وزیر در این سلول به حل مسئله منجر نشود، آن را از صفحه حذف می‌کنیم
        Board[i][col] = 0

# اگر هیچ یک از سلول‌ها منجر به حل مسئله نشود
    Return False

Def solve_n_queens(n):
# ایجاد صفحه شطرنج خالی
    Board = [[0 for _ in range(n)] for _ in range(n)]

# حل مسئله با فراخوانی اولیه از ستون اول
    If not solve_n_queens_util(board, 0, n):
        Print("هیچ راه حلی وجود ندارد.")
        Return False

# نمایش جواب
    For i in range(n):
        For j in range(n):
            Print(board[i][j], end=" ")
        Print()
    Return True
```

برای حل مسئله 8 وزیر #n=8 تابع را فراخوانی می‌کنیم با

Solve_n_queens(8)