```
# Function to check if two queens threaten each other or not
def isSafe(mat, r, c):

    # return false if two queens share the same column
    for i in range(r):
        if mat[i][c] == 'Q':
            return False

    # return false if two queens share the same `` diagonal
    (i, j) = (r, c)
    while i >= 0 and j >= 0:
        if mat[i][j] == 'Q':
            return False
        i = i - 1
        j = j - 1

    # return false if two queens share the same `/` diagonal
    (i, j) = (r, c)
    while i >= 0 and j < len(mat):
        if mat[i][j] == 'Q':
            return False
        i = i - 1
        j = j + 1

    return True
```

```
def printSolution(mat):
    for r in mat:
        print(str(r).replace(',', '').replace('\'', ''))
    print()


def nQueen(mat, r):

    # if `N` queens are placed successfully, print the solution
    if r == len(mat):
        printSolution(mat)
        return

    # place queen at every square in the current row `r`
    # and recur for each valid movement
    for i in range(len(mat)):

        # if no two queens threaten each other
        if isSafe(mat, r, i):
            # place queen on the current square
            mat[r][i] = 'Q'

            # recur for the next row
```

```
            nQueen(mat, r + 1)

            # backtrack and remove the queen from the current square
            mat[r][i] = 'â€“'


if __name__ == '__main__':

    # `N Ã— N` chessboard
    N = 8

    # `mat[][]` keeps track of the position of queens in
    # the current configuration
    mat = [['â€“' for x in range(N)] for y in range(N)]

    nQueen(mat, 0)
```