In [1]:

```
from __future__ import print_function

N = 8

def printSolution(board):
    for i in range(N):
        for j in range(N):
            print(board[i][j], end = " ")
        print()

def isSafe(row, col, slashCode, backslashCode,
           rowLookup, slashCodeLookup,
                       backslashCodeLookup):
    if (slashCodeLookup[slashCode[row][col]] or
        backslashCodeLookup[backslashCode[row][col]] or
        rowLookup[row]):
        return False
    return True

def solveNQueensUtil(board, col, slashCode, backslashCode,
                     rowLookup, slashCodeLookup,
                     backslashCodeLookup):

    if(col >= N):
        return True
    for i in range(N):
        if(isSafe(i, col, slashCode, backslashCode,
                  rowLookup, slashCodeLookup,
                  backslashCodeLookup)):

            board[i][col] = 1
            rowLookup[i] = True
            slashCodeLookup[slashCode[i][col]] = True
            backslashCodeLookup[backslashCode[i][col]] = True

            if(solveNQueensUtil(board, col + 1,
                                slashCode, backslashCode,
                                rowLookup, slashCodeLookup,
                                backslashCodeLookup)):
                return True

            board[i][col] = 0
            rowLookup[i] = False
            slashCodeLookup[slashCode[i][col]] = False
            backslashCodeLookup[backslashCode[i][col]] = False

    return False

def solveNQueens():
    board = [[0 for i in range(N)]
                for j in range(N)]

    # helper matrices
    slashCode = [[0 for i in range(N)]
                    for j in range(N)]
    backslashCode = [[0 for i in range(N)]
                        for j in range(N)]

    # arrays to tell us which rows are occupied
```

```
    rowLookup = [False] * N

    # keep two arrays to tell us
    # which diagonals are occupied
    x = 2 * N - 1
    slashCodeLookup = [False] * x
    backslashCodeLookup = [False] * x

    # initialize helper matrices
    for rr in range(N):
        for cc in range(N):
            slashCode[rr][cc] = rr + cc
            # DIAGONAL CONDITION
            backslashCode[rr][cc] = rr - cc + 7

    if(solveNQueensUtil(board, 0, slashCode, backslashCode,
                        rowLookup, slashCodeLookup,
                        backslashCodeLookup) == False):
        print("Solution does not exist")
        return False

    # solution found
    printSolution(board)
    return True


solveNQueens()
```

```
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
```

Out[1]:

```
True
```

In [ ]: