BRANCH AND BOUND

```
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] = "Q"
            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] = "-"
            rowLookup[i] = False
            slashCodeLookup[slashCode[i][col]] = False

            backslashCodeLookup[backslashCode[i][col]] = False


    return False


def solveNQueens():

    board = [["-" for _ in range(N)] for _ in range(N)]

    slashCode = [["-" for _ in range(N)] for _ in range(N)]

    backslashCode = [["-" for _ in range(N)] for _ in range(N)]

    rowLookup = [False] * N

    x = 2 * N - 1

    slashCodeLookup = [False] * x

    backslashCodeLookup = [False] * x


    for rr in range(N):

        for cc in range(N):

            slashCode[rr][cc] = rr + cc

            backslashCode[rr][cc] = rr - cc + N - 1


    if not solveNQueensUtil(board, 0, slashCode, backslashCode, rowLookup, slashCodeLookup,
backslashCodeLookup):

        print("Solution does not"-" exist")

        return False


    printSolution(board)

    return True


# Prompt the user to enter the board size

N = int(input("Enter the board size: "))

solveNQueens()
```