In [4]:

```python
import random
# n=8
#n-queen using hill-cimbing method
N=8

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


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


def copyState(state1, state2):
    for i in range(N):
        state1[i]= state2[i]


def generateBoard(board, State):
    fill(board)
    for i in range(N):
        board[state[i]][i]= 1

def compareState(state1, state2):
    for i in range(N):
            if(state1[i]!=state2[i]):
                return False
    return True

def calculateObjective(board, state):
    cnt=0
    row=0
    col=0
    for i in range(N):
        row= state[i]
        col= i-1
        while(col>=0):
            if(board[row][col] == 1):
                cnt+=1
            col-=1

        row= state[i]
        col= i+1
        while(col<N):
            if(board[row][col] == 1):
                cnt+=1
            col+=1

        row= state[i]-1
        col = i-1
        while (col>= 0 and row>=0):
            if(board[row][col] == 1):
                cnt+=1
```

```
            col-=1
            row-=1

        row = state[i]+1
        col= i+1
        while (col < N and row < N):
            if(board[row][col] == 1):
                cnt+=1
            col+=1
            row+=1

        row = state[i]+1
        col= i-1
        while (col>=0 and row < N):
            if(board[row][col] == 1):
                cnt+=1
            col-=1
            row+=1


        row = state[i]-1
        col= i+1
        while (col < N and row>=0):
            if(board[row][col] == 1):
                cnt+=1
            col+=1
            row-=1

    return cnt/2


def getNeighbour(board, state):
    opState= [0 for i in range(N)]
    opBoard=[[0 for i in range(N)] for j in range(N)]
    copyState(opState, state)
    generateBoard(opBoard, opState)

    opObjective= calculateObjective(opBoard, opState)

    neighbourState= [0 for i in range(N)]
    neighbourBoard=[[0 for i in range(N)] for j in range(N)]
    copyState(neighbourState, state)
    generateBoard(neighbourBoard, neighbourState)

    for i in range(N):
        for j in range(N):
            if(j!= state[i]):
                neighbourState[i] = j
                neighbourBoard[neighbourState[i]][i]= 1
                neighbourBoard[state[i]][i]= 0

                temp= calculateObjective(neighbourBoard, neighbourState)

                if(temp< opObjective):
                    opObjective= temp
                    copyState(opState, neighbourState)
                    generateBoard(opBoard, opState)


                neighbourBoard[neighbourState[i]][i]= 0
                neighbourState[i] = state[i]
```

```
                neighbourBoard[state[i]][i]= 1

    copyState(state, opState)
    generateBoard(board, state)


def hillCimbing(board, state):
    neighbourState= [0 for i in range(N)]
    neighbourBoard=[[0 for i in range(N)] for j in range(N)]
    copyState(neighbourState, state)
    generateBoard(neighbourBoard, neighbourState)

    while(True):
        copyState(state, neighbourState)
        generateBoard(board, state)
        getNeighbour(neighbourBoard, neighbourState)
        if(compareState(state, neighbourState)):
            printBoard(board)
            break
        if(calculateObjective(board, state)==calculateObjective(neighbourBoard
            neighbourState[random.randrange(0, N)]= random.randrange(0, 8)
            generateBoard(neighbourBoard, neighbourState)




state= [0 for i in range(N)]
board=[[0 for i in range(N)] for j in range(N)]

#configure randomly
for i in range(0, N):
    state[i]= random.randrange(0, 8)
    board[state[i]][i]= 1

hillCimbing(board, state)

```

```
1 0 0 0 0 0 0 0

0 0 0 1 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 1 0 1

0 1 0 0 0 0 0 0

0 0 0 0 1 0 0 0

0 0 0 0 0 0 0 0
```