```
from random import randint
N = 8

# A utility function that configures
# the 2D array "board" and
# array "state" randomly to provide
# a starting point for the algorithm.
def configureRandomly(board, state):

# Iterating through the
# column indices
for i in range(N):
    # Getting a random row index
    state[i] = randint(0, 100000) % N

# Placing a queen on the
# obtained place in
# chessboard.
board[state[i]][i] = 1
```

```
def printBoard(board):
    print("Solution Found!")
    print("Name: Vyom Gupta")
    print("USN: 1BM22CS333")
    print("Board Configuration:")
    for i in range(N):
        print(*board[i])

# A utility function that prints
# the array "state".
def printState(state):
    print(*state)

# A utility function that compares
# two arrays, state1 and state2 and
# returns True if equal
# and False otherwise.
def compareStates(state1, state2):
    for i in range(N):
        if (state1[i] != state2[i]):
            return False
    return True

# A utility function that fills
# the 2D array "board" with
# values "value"
```

```
def fill(board, value):
    for i in range(N):
        for j in range(N):
        board[i][j] = value

# This function calculates the
# objective value of the
# state(queens attacking each other)
# using the board by the
# following logic.
def calculateobjective(board, state):
    attacking = 0

# Checking each queen
for i in range(N):
    row = state[i]
    col = i - 1

# Check for Left attacks (same row)
    while col >= 0 and board[row][col] != 1:
        col -= 1
    if col >= 0 and board[row][col] != 1:
        attacking += 1

# Check for right attacks (same row)
    row = state[i]
    col = i + 1

while col < N and board[row][col] != 1:
    col += 1
    if col < N and board[row][col] == 1:</pre>
```

```
# Diagonal Left-up
row = state[i] - 1
col = i - 1
while col >= 0 and row >= 0 and board[row][col] != 1:
    col -= 1
    row -= 1
if col >= 0 and row >= 0 and board[row][col] == 1:
    attacking += 1

# Diagonal right-down
row = state[i] + 1
col = i + 1
while col < N and row < N and board[row][col] != 1:
    col += 1
    row += 1
if col < N and row < N and board[row][col] == 1:
    attacking += 1

# Diagonal Left-down
row = state[i] + 1
col = i - 1
while col >= 0 and row < N and board[row][col] != 1:
    col -= 1
    row += 1
if col >= 0 and row < N and board[row][col] != 1:
    row += 1
if col >= 0 and row < N and board[row][col] == 1:
    attacking += 1

# Diagonal right-up
row = state[i] - 1
col = i + 1</pre>
```

```
fill(board, 0)
generateBoard(board, state)

def hillClimbing(board, state):
    neighbourBoard = [[0 for _ in range(N)] for _ in range(N)]
    neighbourState = [0 for _ in range(N)]

copyState(neighbourState, state)
generateBoard(neighbourBoard, neighbourState)

while True:
    copyState(state, neighbourState)
generateBoard(board, state)

getNeighbour(neighbourBoard, neighbourState)

if compareStates(state, neighbourState):
    printBoard(board)
    break

elif calculateObjective(board, state) == calculateObjective(neighbourBoard, neighbourState):
    neighbourState[randint(0, 100000) % N] = randint(0, 100000) % N
generateBoard(neighbourBoard, neighbourState)

# Driver code
state = [0] * N
board = [[0 for _ in range(N)] for _ in range(N)]
configureRandomly(board, state)
```

```
Solution Found!
Name: Vyom Gupta
USN: 1BM22CS333
Board Configuration:
O
 0 0 0 0 1 0 0
O
 0 0 1 0 0 0 0
0
 1 0 0 0 0 0 0
0 0 0 0 0 0 0 1
O
 0 0 0 1 0 0 0
0
 0 0 0 0 0 1 0
1
 0 0 0 0 0 0 0
0
 O
    1 0 0 0 0 0
```

Consider 1320 Consid					
Consider 1320 Consid	A sus- even	03	-	10,	03
$\begin{array}{c ccccccccccccccccccccccccccccccccccc$		and the same	-	Q,	Q ₂
$\begin{array}{c ccccccccccccccccccccccccccccccccccc$	10.	- OKTORES		Walter To	
Consider 1320 Consid		0	0	1600	
consider 1320 Q Q Q Q Q Q Q Q Q Q Q Q Q Q Q Q Q Q Q		23	6		cost = 1
Consider 1320 Q Q Q Q Q Q Q Q Q Q Q Q Q Q Q Q Q Q Q	THE RESERVE AND ADDRESS OF THE PARTY NAMED IN	100	0, - 02	Service .	
Consider 1320 Q. Q. Q. O. 3 1 3 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	.10.1		010	Old Dis	
Q, Q, Q ₃ 3 2 0 Q, Q ₃ 3 1 3 0 Q, Q ₃ 3 1 0 Q, Q ₄ 3 1 0 Q, Q ₄ 2 2 1 Q, Q ₅ 1 2 3 D Q, Q ₆ 1 2 3 D Q, Q ₇ 1 0 3 2 1 Q, Q ₇ 1 0 3 2 1 Q, Q ₇ 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	3 8 5 4		3	2	de la compania
Q, Q, Q ₃ 3 2 0 Q, Q ₃ 3 1 3 0 Q, Q ₃ 3 1 0 Q, Q ₄ 3 1 0 Q, Q ₄ 2 2 1 Q, Q ₅ 1 2 3 D Q, Q ₆ 1 2 3 D Q, Q ₇ 1 0 3 2 1 Q, Q ₇ 1 0 3 2 1 Q, Q ₇ 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	anni'm		· · · · · · · · · · · · · · · · · · ·		
3 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	corsis.	-1320	- d	10103	
3 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0		0	1 20200	13000	
3 1 3 0 Q. 2 3 1 0 3 2 1 10 3 2 1 10 3 2 1 10 3 2 2 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	Q,	Q, B	2 03	2	1110
2 3 1 0 0 3 2 1 1 6 3 2 1 7 0 2 1 7 0 2)	3 2	- 0	7 2	12/03
2 3 1 0 0 3 2 1 1 6 3 2 1 7 0 2 1 7 0 2	2	1	20		100
1 2 3 D 1 3 2 1 6 3 2 1 7 D 2 1 7 D 2 1 7 D 0 D D D D D D D D D D D D D D D D D			1 0		
10 3 2 1 1 1 2 3 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	2	3	1 0		1-61
1 6 3 2 1 7 0 2 Put. Quenz 1, Empty 20 0 0 0 0 0	0	3 2	_ 1	0	1 10
1 6 3 2 1 7 0 2 Put. Quenz 1, Empty 20 0 0 0 0 0	(10	, 3	0	no ma	1 to proces
1702 put. Quenzt, Empty=0		1	0	12 P	0000
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0		6 3	04	01	
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0		2 0	2	3	0000
000000		5	A Pa PM	100	001
000000			0 0	0	
000000	100	त .	Burghyz	0	0000
000000	out. One	W5+ 1	10	10	0
000000	0 0	0	0	,	-
100000			10	00	0
1000000	0 0		7	-	0
1000010		A 6	0	0 0	
	0	U	1	0 1	0
		2			2 .0