DATE:

N-QUEENS PROBLEM

EX: 01

Aim:

To implement a python program for N Queens - Backtracking.

Algorithm:

1. Initialize an NXN chessboard with all entries as 0.
2. check safety (issues)
   Ensure no queens threaten the position (row, col) by checking the row, and diagonals to the left.
3. Recursive placement (solve NQUtil):
   - for each column, try placing a queen in each row.
   - if safe, place the queens and recurse to the next column.
   - If no solution, backtrack (remove queen and try next row).
4. Base case: if all queens are placed, return True.
5. Solve: start from the first column, recursively place queens and print the solution is found.

Program:

```
global N
N=4

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

def isSafe(board, row, col):
    for i in range(col):
        if board[row][i] == 1:
            return False
    for i,j in zip(range(row,-1,-1), range
            (col,-1,-1)):
        if board[i][j] == 1:
            return False
    for i,j in zip(range(row,N,1), range(
            col,-1,-1)):
        if board[i][j] == 1:
            return False
    return True

def solveNQUtil(board, col):
    if col >= N:
        return True
    return False
```

```
def solveNQ():
    board = [ [0,0,0,0],
              [0,0,0,0],
              [0,0,0,0],
              [0,0,0,0]
            ]
    if solveNQUtil(board, 0) == False:
        print("Solution does not exist")
        return False

    printSolution(board)
    return True

solveNQ()
```

output :

```
0 0 1 0
1 0 0 0
0 0 0 1
0 1 0 0
```

Result:

N-Queens problem ~~for~~ using backtracking is successfully implemented.