TASK-8 N-queen problem using backtracking algorithm

PROGRAM

```
# Python3 program to solve N Queen
# Problem using backtracking
global N
N = 4
def printSolution(board):
for i in range(N):
for j in range(N):
if board[i][j] == 1:
print("Q",end=" ")
else:
print(".",end=" ")
print()
def isSafe(board, row, col):
# Check this row on left side
for i in range(col):
if board[row][i] == 1:
return False
# Check upper diagonal on left side
for i, j in zip(range(row, -1, -1),
range(col, -1, -1)):
if board[i][j] == 1:
return False
# Check lower diagonal on left side
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):
# Base case: If all queens are placed
# then return true
if col >= N:
return True
# Consider this column and try placing
# this queen in all rows one by one
for i in range(N):
if isSafe(board, i, col):
# Place this queen in board[i][col]
board[i][col] = 1
if solveNQUtil(board, col + 1) == True:
return True
board[i][col] = 0
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
# Driver Code
if __name__ == '__main__':
solveNQ()
```

OUTPUT

```
> & C:/Users/preet/AppData/Local/Programs/Python/Python314/python.exe "c:/Users/preet/Downloads/python tut/task 8.py"
. . Q .
Q . . .
. . . Q
. Q . .
PS C:\Users\preet\Downloads\python tut>
```