N - Queens Problem

Aim:

To solve the N - Queens problem by identifying all valid configurations of or finding atleast one solution, depending on the requirement.

Algorithm:

1. Start
2. ~~For each colu~~
2. Create an N×N board with all positions set to 0 (empty)
3. For each column, try placing a queen in each row
4. Check if the position is safe using the isSafe() function
5. If placing the queen is safe, mark the position with a 1
6. Recursively move to the next column and repeat
7. If no valid placement is found, backtrack by removing the queen, and by a new position.
8. If queens are placed in all columns, print the board.
9. If no placement is possible print message
10. Stop

```
def printSolution(board):
    for i in range(N):
        for j in range(N):
            if board[i][j] == 1:
                print("1", end=' ')
            else:
                print("0", 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

    for i in range(N):
        if isSafe(board, i, col):
            board[i][col] = 1
            if SolveNQUtil(board, col+1):
                return True
            board[i][col] = 0
    return False
```

```
def SolveNQ():
    global N
    N = int(input())
    board [[0 for i in range(N)] for j in range(N)]
    if SolveNQUtil(board, 0) == False:
        print("Solution does not exist")
        return False
        printSolution(board)
        return True

SolveNQ()
```

Output:

```
Enter the number : 3
Solution does not exist
Enter the number : 4
_ _ Q _
Q _ _ _
_ _ _ Q
_ Q _ _
```

Result:

The program has been executed successfully and the output has been verified.