SRM INSTITUTE OF SCIENCE & TECHNOLOGY

DEPARTMENT OF NETWORKING & COMMUNICATIONS

18CSC305J-ARTIFICIAL INTELLIGENCE

SEMESTER – 6

BATCH-1

REGISTRATION NUMBER	**RA1911003011013**
NAME	**Shivank Arora**

B.Tech- CSE / CC, Third Year (Section: H2)

Faculty In charge: Dr. S. Prabakeran, B.Tech, M.E, PhD
Assistant Professor
School of Computing - Department of Networking and Communications

Year 2021-2022 / Even Semester

INDEX

Ex No	DATE	Title	Page No	Marks
1	05-01-2022	Toy Problem: (4 queens problem)		

Exercise: 1

Date : 05-01-2022

TOY PROBLEM(4 queens problem)

Problem Statement : The *4-Queens Problem* consists in placing four queens on a 4 x 4 chessboard so that no two queens can capture each other. That is, no two queens are allowed to be placed on the same row, the same column or the same diagonal

Tool : jupyter notebook

Programming code :

```
def printSol(board):
  for i in range(4):
    for j in range(4):
      print (board[i][j])

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, 4, 1), range(col, -1, -1)):
      if board[i][j]==1:
        return False
    return True


def sol(board, col):
  if col>=4:
    return True
  for i in range(4):
    if isSafe(board, i, col):
      board[i][col]=1
      if sol(board, col+1)==True:
        return True
      board[i][col]=0
  return False
def solNQueens():
    board = [[0, 0, 0, 0],
        [0, 0, 0, 0],
        [0, 0, 0, 0],
        [0, 0, 0, 0]]

    if sol(board, 0)==False:
        print("no solution")
        return False
    printSol(board)
    return True
solNQueens()
```

Output screen shots :

In [1]:

```
def printSol(board):
  for i in range(4):
    for j in range(4):
      print (board[i][j])

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, 4, 1), range(col, -1, -1)):
      if board[i][j]==1:
        return False
    return True


def sol(board, col):
  if col>=4:
    return True
  for i in range(4):
    if isSafe(board, i, col):
      board[i][col]=1
      if sol(board, col+1)==True:
        return True
      board[i][col]=0
  return False
def solNQueens():
    board = [[0, 0, 0, 0],
        [0, 0, 0, 0],
        [0, 0, 0, 0],
        [0, 0, 0, 0]]

    if sol(board, 0)==False:
        print("no solution")
        return False
    printSol(board)
    return True
solNQueens()
```

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

Result : Queens are placed on 4 x 4 chessboard so that no two queens can capture each other