Practical No. 4

‘N’ Queens Problem

**Q.1]** Write a Python program for implementing the ‘N’ Queens problem.

**Code:**

import copy as cp

**class** nQueens:

sz = 1

board = []

total = 1

**def** initInput(self):

self.sz = int(input("Enter the size of the chessboard: "))

self.board = [[" " for i in range(self.sz)] for j in range(self.sz)]

**def** printBoard(self, brd):

print("Board: ",self.total)

for i in range(self.sz):

for j in range(self.sz):

print(brd[i][j]+" |",end=" ")

print("\n")

print("\n")

self.total += 1

*#for i in self.board:*

*# print(i)*

**def** checkIsValid(self, board, row, col):

for i in range(col):

if board[row][i] == "q":

return False

for i, j in zip(range(row, -1, -1),range(col, -1, -1)):

if board[i][j] == "q":

return False

for i, j in zip(range(row, self.sz, 1),range(col, -1, -1)):

if board[i][j] == "q":

return False

*#above code is only checking above from current pos because board is not filled below*

*#so need to check below board from current pos*

return True

**def** doAlgo(self, col, brd):

if (col >= self.sz):

self.printBoard(brd)

return True

for i in range(self.sz):

if (self.checkIsValid(cp.deepcopy(brd), i, col) == True):

brd[i][col] = "q"

if (self.doAlgo(col+1, cp.deepcopy(brd)) == False):

brd[i][col] = " " *#if no possible placements for a queen in next row then cur placement is invalid*

*#else if valid then algo will keep recursively running declare in the 'if' statement*

*#Below will return false when no placement possible for the queen*

if (col > 0): return False *#this if is not necessay just there for saying this func doesn't returns anything*

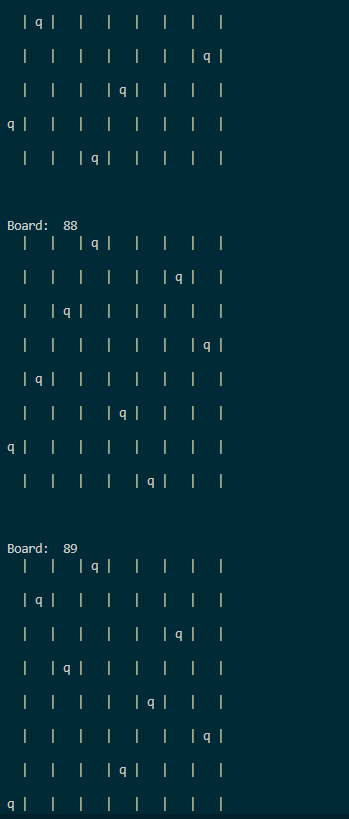
nQueensSol = nQueens()

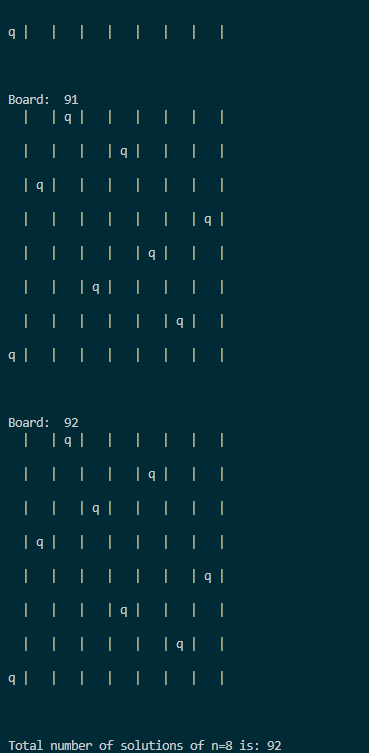
nQueensSol.initInput()

nQueensSol.doAlgo(0, nQueensSol.board)

print(**f**"Total number of solutions of n={nQueensSol.sz} is: {nQueensSol.total-1}")

**Output:**

****

****