Name:

Roll No.:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Aim: Use back-tracking to place remaining Queens to generate final 8-queen's Matrix using Python.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

def Permute(queens, row):

for i in range(8):

queens[row] = i

if Fine(queens, row):

if row == 7:

print(queens)

globals()["solutions"] = globals()["solutions"] + 1

else:

Permute(queens, row+1)

def Fine(queens, row):

c = 0

derga = True

for i in range(row):

c, cur, oth = c+1, queens[row], queens[row-i-1]

if (cur == oth) or (cur-c == oth) or (cur+c == oth):

derga = False

break

return(derga)

globals()["solutions"] = 0

queens = [20, 20, 20, 20, 20, 20, 20, 20]

for i in range(8):

queens[0] = i

Permute(queens, 1)

print(solutions)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Output: