**AI PRACTICAL ASSIGNMENT – 04**

class NQueensProblem:

def \_init\_(self, n):

self.queens = [0] \* n

self.numSolutions = 0

def solve(self):

self.solve\_helper(0)

def solve\_helper(self, row):

if row == len(self.queens):

self.numSolutions += 1

self.print\_solution()

else:

for col in range(len(self.queens)):

self.queens[row] = col

if self.is\_valid(row, col):

self.solve\_helper(row + 1)

def is\_valid(self, row, col):

for i in range(row):

diff = abs(self.queens[i] - col)

if diff == 0 or diff == row - i:

return False

return True

def print\_solution(self):

if self.numSolutions == 1:

print("Solution: ", end="")

for i in range(len(self.queens)):

print(self.queens[i], end=" ")

print()

print("The Matrix Representation:")

arr = [[0] \* len(self.queens) for \_ in range(len(self.queens))]

for i in range(len(self.queens)):

for j in range(len(self.queens)):

if j == self.queens[i]:

arr[i][j] = 1

for i in range(len(self.queens)):

for j in range(len(self.queens)):

print(arr[i][j], end=" ")

print()

if \_name\_ == "\_main\_":

n = int(input("Enter N Queens Problem: "))

NQueensProblem = NQueensProblem(n)

NQueensProblem.solve()

**OUTPUT**

Enter N Queens Problem: 4

Solution: 1 3 0 2

The Matrix Representation:

0 1 0 0

0 0 0 1

1 0 0 0

0 0 1 0

Enter N Queens Problem: 8

Solution: 0 4 7 5 2 6 1 3

The Matrix Representation: 1 0 0 0 0 0 0 0

0 0 0 0 1 0 0 0

0 0 0 0 0 0 0 1

0 0 0 0 0 1 0 0

0 0 1 0 0 0 0 0

0 0 0 0 0 0 1 0

0 1 0 0 0 0 0 0

0 0 0 1 0 0 0 0