**Name-**Pawar Aadesh

**Roll no-**33

**Problem Statement-**Implement a solution for a Constraint Satisfaction Problem using Branch and Bound and Backtracking for n-queens problem or a graph coloring problem.

class NQueens:

def \_\_init\_\_(self, n):

self.n = n

self.solutions = []

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

for r in range(row):

if board[r] == col or board[r] - r == col - row or board[r] + r == col + row:

return False

return True

def solve(self, board, row):

if row == self.n:

self.solutions.append(board[:])

return

for col in range(self.n):

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

board[row] = col

self.solve(board, row + 1)

def print\_board(self, board):

for row in range(self.n):

row\_representation = ""

for col in range(self.n):

if board[row] == col:

row\_representation += "Q "

else:

row\_representation += ". "

print(row\_representation)

print("\n")

def get\_solutions(self):

board = [-1] \* self.n

self.solve(board, 0)

return self.solutions

# Example usage for N = 4

n\_queens = NQueens(4)

solutions = n\_queens.get\_solutions()

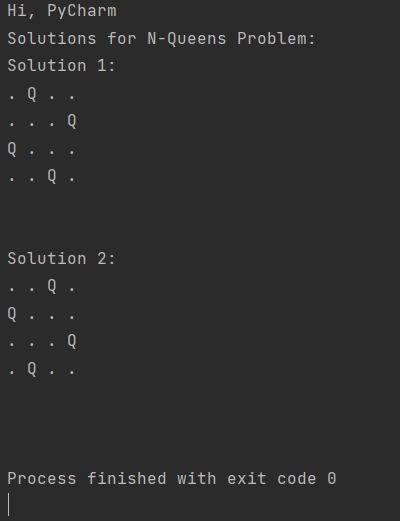
print("Solutions for N-Queens Problem:")

for idx, sol in enumerate(solutions):

print(f"Solution {idx + 1}:")

n\_queens.print\_board(sol)

OUTPUT

****