**Exp :5 Write the python program for Missionaries Cannibal problem.**

**Input:**

from collections import deque

start\_state = (3, 3, 1)

goal\_state = (0, 0, 0)

moves = [(2, 0), (1, 0), (1, 1), (0, 1), (0, 2)]

def is\_valid(state):

    m\_left, c\_left, \_ = state

    m\_right = 3 - m\_left

    c\_right = 3 - c\_left

    if (m\_left < 0 or m\_left > 3 or c\_left < 0 or c\_left > 3):

        return False

    if (m\_right < 0 or m\_right > 3 or c\_right < 0 or c\_right > 3):

        return False

    if (m\_left > 0 and c\_left > m\_left):

        return False

    if (m\_right > 0 and c\_right > m\_right):

        return False

    return True

def get\_successors(state):

    successors = []

    m, c, boat = state

    for m\_move, c\_move in moves:

        if boat == 1:

            new\_state = (m - m\_move, c - c\_move, 0)

        else:

            new\_state = (m + m\_move, c + c\_move, 1)

        if is\_valid(new\_state):

            successors.append(new\_state)

    return successors

def bfs():

    queue = deque()

    queue.append((start\_state, [start\_state]))

    visited = set()

    while queue:

        current\_state, path = queue.popleft()

        if current\_state == goal\_state:

            return path

        for next\_state in get\_successors(current\_state):

            if next\_state not in visited:

                visited.add(next\_state)

                queue.append((next\_state, path + [next\_state]))

    return None

solution = bfs()

if solution:

    print("Solution found:")

    for step in solution:

        m, c, b = step

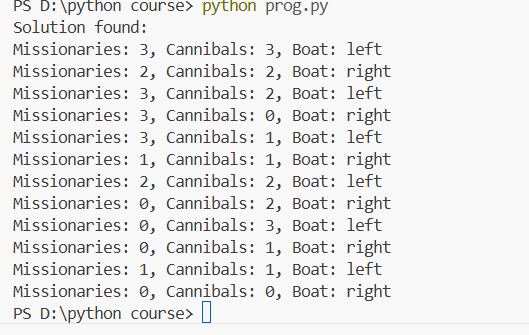
        side = 'left' if b == 1 else 'right'

        print(f"Missionaries: {m}, Cannibals: {c}, Boat: {side}")

else:

    print("No solution found.")

**output:**

****