from collections import deque

# Function to check if the state is valid

def is\_valid(state):

m1, c1, boat, m2, c2 = state

if m1 < 0 or c1 < 0 or m2 < 0 or c2 < 0:

return False

if (m1 > 0 and m1 < c1) or (m2 > 0 and m2 < c2): # Cannibals can't outnumber missionaries

return False

return True

# Function to solve the problem using BFS

def missionaries\_cannibals():

start\_state = (

3, 3, 1, 0, 0) # (missionaries on left, cannibals on left, boat, missionaries on right, cannibals on right)

goal\_state = (0, 0, 0, 3, 3) # Goal state (all missionaries and cannibals on right)

queue = deque([(start\_state, [])])

visited = set()

while queue:

state, path = queue.popleft()

if state in visited:

continue

visited.add(state)

if state == goal\_state:

return path + [state]

m1, c1, boat, m2, c2 = state

# Possible moves

if boat == 1: # Boat on left side

moves = [

(m1 - 2, c1, 0, m2 + 2, c2), # Two missionaries

(m1, c1 - 2, 0, m2, c2 + 2), # Two cannibals

(m1 - 1, c1 - 1, 0, m2 + 1, c2 + 1), # One missionary and one cannibal

(m1 - 1, c1, 0, m2 + 1, c2), # One missionary

(m1, c1 - 1, 0, m2, c2 + 1) # One cannibal

]

else: # Boat on right side

moves = [

(m1 + 2, c1, 1, m2 - 2, c2), # Two missionaries

(m1, c1 + 2, 1, m2, c2 - 2), # Two cannibals

(m1 + 1, c1 + 1, 1, m2 - 1, c2 - 1), # One missionary and one cannibal

(m1 + 1, c1, 1, m2 - 1, c2), # One missionary

(m1, c1 + 1, 1, m2, c2 - 1) # One cannibal

]

for move in moves:

if is\_valid(move):

queue.append((move, path + [state]))

return None

# Running the solver function

solution = missionaries\_cannibals()

if solution:

for step in solution:

print(step)

else:

print("No solution found.")

Output:  
