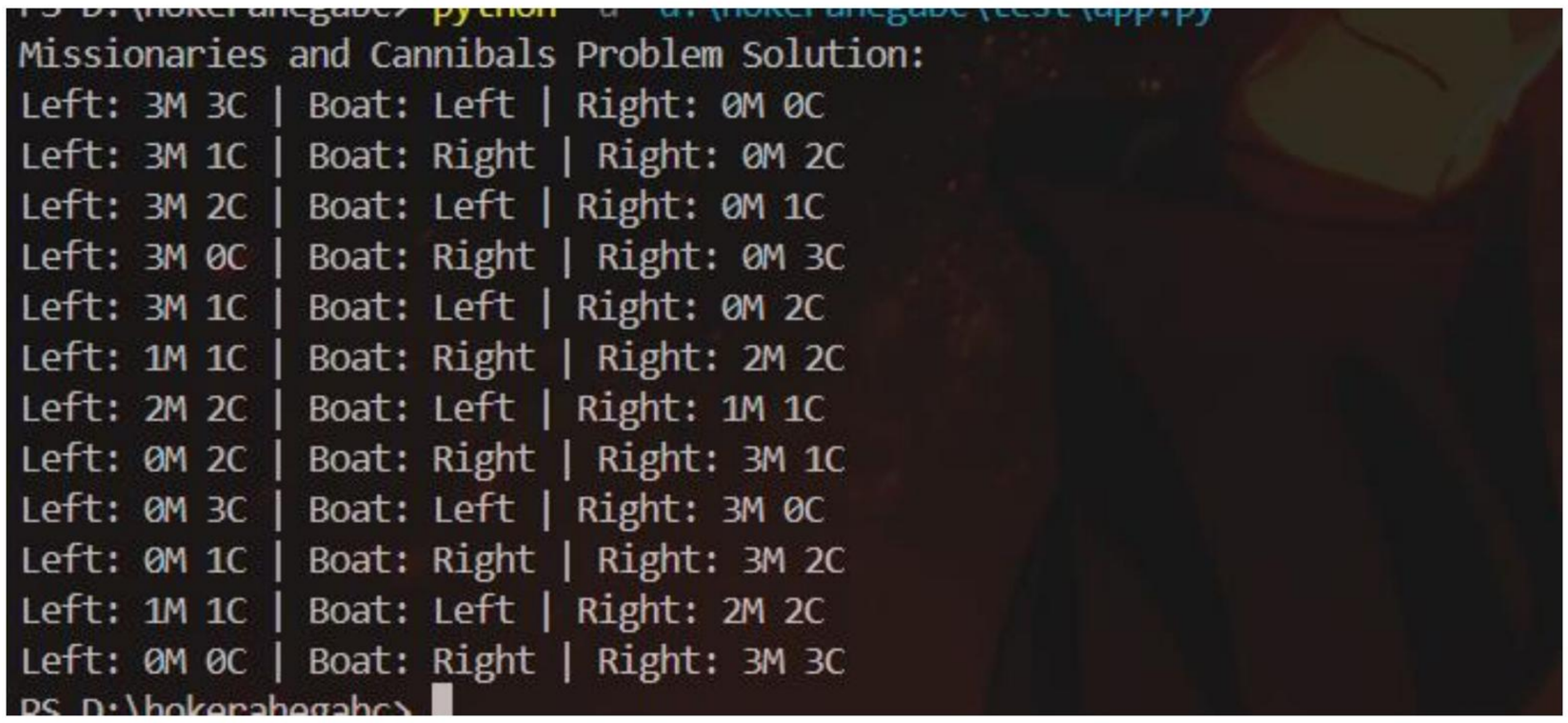
Name :- Aditya Kumar Singh

Reg No. :- RA2311003030916

Section :- O1

Q1. The Missionaries and cannibal problem.

```python
from collections import deque

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):
        return False
    return True

def get_next_states(state):
    m1, c1, boat, m2, c2 = state
    moves = [(1, 0), (2, 0), (0, 1), (0, 2), (1, 1)]
    next_states = []

    if boat == 1:  # Boat on left side
        for m, c in moves:
            new_state = (m1 - m, c1 - c, 0, m2 + m, c2 + c)
            if is_valid(new_state):
                next_states.append(new_state)
    else:  # Boat on right side
        for m, c in moves:
            new_state = (m1 + m, c1 + c, 1, m2 - m, c2 - c)
            if is_valid(new_state):
                next_states.append(new_state)

    return next_states

def bfs():
    start = (3, 3, 1, 0, 0)  # (M_left, C_left, Boat, M_right, C_right)
    goal = (0, 0, 0, 3, 3)
    queue = deque([(start, [])])
    visited = set([start])

    while queue:
        state, path = queue.popleft()
        if state == goal:
            return path + [state]

        for next_state in get_next_states(state):
            if next_state not in visited:
                queue.append((next_state, path + [state]))
                visited.add(next_state)

    return None

def print_solution(solution):
    if not solution:
        print("No solution found!")
        return

    print("Missionaries and Cannibals Problem Solution:")
    for step in solution:
        m1, c1, boat, m2, c2 = step
        boat_side = "Left" if boat == 1 else "Right"
        print(f"Left: {m1}M {c1}C | Boat: {boat_side} | Right: {m2}M {c2}C")

solution = bfs()
print_solution(solution)
```


Missionaries and Cannibals Problem Solution:
Left: 3M 3C | Boat: Left | Right: 0M 0C
Left: 3M 1C | Boat: Right | Right: 0M 2C
Left: 3M 2C | Boat: Left | Right: 0M 1C
Left: 3M 0C | Boat: Right | Right: 0M 3C
Left: 3M 1C | Boat: Left | Right: 0M 2C
Left: 1M 1C | Boat: Right | Right: 2M 2C
Left: 2M 2C | Boat: Left | Right: 1M 1C
Left: 0M 2C | Boat: Right | Right: 3M 1C
Left: 0M 3C | Boat: Left | Right: 3M 0C
Left: 0M 1C | Boat: Right | Right: 3M 2C
Left: 1M 1C | Boat: Left | Right: 2M 2C
Left: 0M 0C | Boat: Right | Right: 3M 3C

Q2. The A* algorithm.

```python
import heapq

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):
        return False
    return True

def heuristic(state):
    return state[3] + state[4]  # People on the right side (goal side)

def get_next_states(state):
    m1, c1, boat, m2, c2 = state
    moves = [(1, 0), (2, 0), (0, 1), (0, 2), (1, 1)]
    next_states = []

    if boat == 1:  # Boat on left
        for m, c in moves:
            new_state = (m1 - m, c1 - c, 0, m2 + m, c2 + c)
            if is_valid(new_state):
                next_states.append(new_state)
    else:  # Boat on right
        for m, c in moves:
            new_state = (m1 + m, c1 + c, 1, m2 - m, c2 - c)
            if is_valid(new_state):
                next_states.append(new_state)
    return next_states

def greedy_bfs():
    start = (3, 3, 1, 0, 0)
    goal = (0, 0, 0, 3, 3)
    queue = []
    heapq.heappush(queue, (heuristic(start), start, []))
    visited = set()

    while queue:
        _, state, path = heapq.heappop(queue)
        if state == goal:
            return path + [state]

        visited.add(state)
        for next_state in get_next_states(state):
            if next_state not in visited:
                heapq.heappush(queue, (heuristic(next_state), next_state, path + [state]))

    return None

solution = greedy_bfs()

if solution:
    for step in solution:
        print(step)
else:
    print("No solution found")
```

```
PS D:\hokeranegabc> python -u "d:\hokeranegabc\test\app.py"
Path found: [(0, 0), (1, 0), (2, 0), (3, 0), (4, 0), (4, 1), (4, 2), (4, 3), (4, 4)]
PS D:\hokeranegabc>
```