```
from collections import deque

class State:
   def __init__(self, missionaries_left, cannibals_left, boat, missionaries_right,
cannibals_right):
       self.missionaries_left = missionaries_left
       self.cannibals_left = cannibals_left
       self.boat = boat
       self.missionaries_right = missionaries_right
       self.cannibals_right = cannibals_right
       self.parent = None

   def is_goal(self):
       return self.missionaries_left == 0 and self.cannibals_left == 0

   def is_valid(self):
       if self.missionaries_left >= 0 and self.missionaries_right >= 0 \
               and self.cannibals_left >= 0 and self.cannibals_right >= 0 \
               and (self.missionaries_left == 0 or self.missionaries_left >=
self.cannibals_left) \
               and (self.missionaries_right == 0 or self.missionaries_right >=
self.cannibals_right):
           return True
       return False

   def __eq__(self, other):
       return self.missionaries_left == other.missionaries_left and \
              self.cannibals_left == other.cannibals_left and \
              self.boat == other.boat and \
              self.missionaries_right == other.missionaries_right and \
              self.cannibals_right == other.cannibals_right

   def __hash__(self):
       return hash((self.missionaries_left, self.cannibals_left, self.boat,
self.missionaries_right, self.cannibals_right))

def successors(current_state):
   children = []
   if current_state.boat == 'left':
       new_state = State(current_state.missionaries_left, current_state.cannibals_left
- 2, 'right',
```

```
                        current_state.missionaries_right,
current_state.cannibals_right + 2)
        if new_state.is_valid():
            new_state.parent = current_state
            children.append(new_state)

        new_state = State(current_state.missionaries_left - 2,
current_state.cannibals_left, 'right',
                        current_state.missionaries_right + 2,
current_state.cannibals_right)
        if new_state.is_valid():
            new_state.parent = current_state
            children.append(new_state)

        new_state = State(current_state.missionaries_left - 1,
current_state.cannibals_left - 1, 'right',
                        current_state.missionaries_right + 1,
current_state.cannibals_right + 1)
        if new_state.is_valid():
            new_state.parent = current_state
            children.append(new_state)

        new_state = State(current_state.missionaries_left, current_state.cannibals_left
- 1, 'right',
                        current_state.missionaries_right,
current_state.cannibals_right + 1)
        if new_state.is_valid():
            new_state.parent = current_state
            children.append(new_state)

        new_state = State(current_state.missionaries_left - 1,
current_state.cannibals_left, 'right',
                        current_state.missionaries_right + 1,
current_state.cannibals_right)
        if new_state.is_valid():
            new_state.parent = current_state
            children.append(new_state)
    else:
        new_state = State(current_state.missionaries_left, current_state.cannibals_left
+ 2, 'left',
                        current_state.missionaries_right,
current_state.cannibals_right - 2)
```

```
        if new_state.is_valid():
            new_state.parent = current_state
            children.append(new_state)

        new_state = State(current_state.missionaries_left + 2,
current_state.cannibals_left, 'left',
                          current_state.missionaries_right - 2,
current_state.cannibals_right)
        if new_state.is_valid():
            new_state.parent = current_state
            children.append(new_state)

        new_state = State(current_state.missionaries_left + 1,
current_state.cannibals_left + 1, 'left',
                          current_state.missionaries_right - 1,
current_state.cannibals_right - 1)
        if new_state.is_valid():
            new_state.parent = current_state
            children.append(new_state)

        new_state = State(current_state.missionaries_left, current_state.cannibals_left
+ 1, 'left',
                          current_state.missionaries_right,
current_state.cannibals_right - 1)
        if new_state.is_valid():
            new_state.parent = current_state
            children.append(new_state)

        new_state = State(current_state.missionaries_left + 1,
current_state.cannibals_left, 'left',
                          current_state.missionaries_right - 1,
current_state.cannibals_right)
        if new_state.is_valid():
            new_state.parent = current_state
            children.append(new_state)

    return children

def breadth_first_search():
    initial_state = State(3, 3, 'left', 0, 0)
    if initial_state.is_goal():
        return initial_state
```

```
    frontier = deque()
    explored = set()
    frontier.append(initial_state)

    while frontier:
        state = frontier.popleft()
        if state.is_goal():
            return state
        explored.add(state)
        children = successors(state)
        for child in children:
            if child not in explored or child not in frontier:
                frontier.append(child)

    return None

def print_solution(solution):
    path = []
    while solution:
        path.append(solution)
        solution = solution.parent
    for t in path[::-1]:
        print(t.missionaries_left, t.cannibals_left, t.boat, t.missionaries_right,
t.cannibals_right)

solution = breadth_first_search()
print_solution(solution)
```