```python
from __future__ import annotations
from typing import List, Optional
from generic_search import bfs, Node, node_to_path

MAX_NUM: int = 3


class MCState:
    def __init__(self, missionaries: int, cannibals: int, boat: bool) -> None:
        self.wm: int = missionaries # 서쪽 강둑에 있는 선교사 수
        self.wc: int = cannibals    # 서쪽 강둑에 있는 식인종 수
        self.em: int = MAX_NUM - self.wm
        self.ec: int = MAX_NUM - self.wc
        self.boat: bool = boat
    # end of __init__

    def __str__(self) -> str:
        return ("서쪽 강둑에는 {}명의 선교사와 {}명의 식인종이 있다.\n"
               "동쪽 강둑에는 {}명의 선교사와 {}명의 식인종이 있다.\n"
               "배는 {}쪽에 있다.")\
            .format(self.wm, self.wc, self.em, self.ec, ("west" if self.boat else "east"))
    # end of __str__

    def goal_test(self) -> bool:
        return self.is_legal and self.em == MAX_NUM and self.ec == MAX_NUM
    # end of goal_test

    @property
    def is_legal(self) -> bool:
        if self.wm < self.wc and self.wm > 0:
            return False

        if self.em < self.wc and self.em > 0:
            return False

        return True
    # end of is_legal

    def successors(self) -> List[MCState]:
        sucs: List[MCState] = []
        if self.boat:
            if self.wm > 1:
                sucs.append(MCState(self.wm - 2, self.wc, not self.boat))
            if self.wm > 0:
                sucs.append(MCState(self.wm - 1, self.wc, not self.boat))
            if self.wc > 1:
                sucs.append(MCState(self.wm, self.wc - 2, not self.boat))
            if self.wc > 0:
                sucs.append(MCState(self.wm, self.wc - 1, not self.boat))
            if (self.wc > 0) and (self.wm > 0):
                sucs.append(MCState(self.wm - 1, self.wc - 1, not self.boat))
        else:
            if self.em > 1:
                sucs.append(MCState(self.wm + 2, self.wc, not self.boat))
            if self.em > 0:
                sucs.append(MCState(self.wm + 1, self.wc, not self.boat))
            if self.ec > 1:
                sucs.append(MCState(self.wm, self.wc + 2, not self.boat))
            if self.ec > 0:
                sucs.append(MCState(self.wm, self.wc + 1, not self.boat))
            if (self.ec > 0) and (self.em > 0):
                sucs.append(MCState(self.em + 1, self.ec + 1, not self.boat))
        return [x for x in sucs if x.is_legal]


def display_solution(path: List[MCState]):
    if len(path) == 0:
        return
    old_state: MCState = path[0]
    print(old_state)
    for current_state in path[1:]:
        if current_state.boat:
            print("{}명의 선교사와 {}명의 식인종이 동쪽 강둑에서 서쪽 강둑으로 갔다.\n"
                  .format(old_state.em - current_state.em, old_state.ec - current_state.ec))
        else:
            print("{}명의 선교사와 {}명의 식인종이 서쪽 강둑에서 동쪽 강둑으로 갔다.\n"
                  .format(old_state.wm - current_state.wm, old_state.wc - current_state.wc))

        print(current_state)
        old_state = current_state


if __name__ == "__main__":
    start: MCState = MCState(MAX_NUM, MAX_NUM, True)
    solution: Optional[Node[MCState]] = bfs(start, MCState.goal_test,
                                            MCState.successors)
    if solution is None:
        print("답을 찾을 수 없습니다.")
    else:
        path: List[MCState] = node_to_path(solution)
        display_solution(path)
```