EXP 5 :-

PROGRAM :-

def is\_safe(state):

m, c, b = state

return (m == 0 or m >= c) and (3 - m == 0 or 3 - m >= 3 - c)

def get\_successors(state):

m, c, b = state

successors = []

if b == 1: # Boat is on the right

if m > 0:

successors.append((m - 1, c, 0))

if m > 1:

successors.append((m - 2, c, 0))

if c > 0:

successors.append((m, c - 1, 0))

if c > 1:

successors.append((m, c - 2, 0))

if m > 0 and c > 0:

successors.append((m - 1, c - 1, 0))

else: # Boat is on the left

if m < 3:

successors.append((m + 1, c, 1))

if m < 2:

successors.append((m + 2, c, 1))

if c < 3:

successors.append((m, c + 1, 1))

if c < 2:

successors.append((m, c + 2, 1))

if m < 3 and c < 3:

successors.append((m + 1, c + 1, 1))

return [s for s in successors if is\_safe(s)]

def bfs():

initial\_state = (3, 3, 1)

goal\_state = (0, 0, 0)

queue = [(initial\_state, [])]

visited = set()

while queue:

state, path = queue.pop(0)

if state in visited:

continue

visited.add(state)

path = path + [state]

if state == goal\_state:

return path

for successor in get\_successors(state):

queue.append((successor, path))

solution = bfs()

for step in solution:

print(step)

OUTPUT :-

