# State representation: (monkey\_pos, monkey\_status, box\_pos, banana\_status)  
initial\_state=('left', 'onfloor', 'middle', 'hasnot')  
  
  
# Actions  
def grasp(state):  
 if state == ('middle', 'onbox', 'middle', 'hasnot'):  
 return ('middle', 'onbox', 'middle', 'has')  
 return None  
  
  
def climb(state):  
 monkey\_pos, monkey\_status, box\_pos, banana\_status=state  
 if monkey\_status == 'onfloor' and monkey\_pos == box\_pos:  
 return (monkey\_pos, 'onbox', box\_pos, banana\_status)  
 return None  
  
  
def push(state, L2):  
 monkey\_pos, monkey\_status, box\_pos, banana\_status=state  
 if monkey\_status == 'onfloor' and monkey\_pos == box\_pos:  
 return (L2, 'onfloor', L2, banana\_status)  
 return None  
  
  
def walk(state, L2):  
 monkey\_pos, monkey\_status, box\_pos, banana\_status=state  
 if monkey\_status == 'onfloor':  
 return (L2, 'onfloor', box\_pos, banana\_status)  
 return None  
  
  
# Goal checking function  
def canget(state):  
 if state[3] == 'has': # banana\_status == 'has'  
 return True, []  
  
 # Define possible actions  
 actions=[  
 ('grasp', grasp),  
 ('climb', climb),  
 ('push\_middle', lambda s: push(s, 'middle')),  
 ('push\_left', lambda s: push(s, 'left')),  
 ('push\_right', lambda s: push(s, 'right')),  
 ('walk\_middle', lambda s: walk(s, 'middle')),  
 ('walk\_left', lambda s: walk(s, 'left')),  
 ('walk\_right', lambda s: walk(s, 'right'))  
 ]  
  
 # Try each action  
 for action\_name, action\_fn in actions:  
 new\_state=action\_fn(state)  
 if new\_state:  
 success, plan=canget(new\_state)  
 if success:  
 return True, [action\_name] + plan  
 return False, []  
  
  
# Find the solution  
success, plan=canget(initial\_state)  
  
if success:  
 print("Plan to get the banana:", plan)  
else:  
 print("No plan found.")