Pgm 1: Implement and Demonstrate Depth First Search Algorithm on Water Jug Problem

```
class WaterJugState:
    def __init__(self, jug1, jug2):
        self.jug1 = jug1
        self.jug2 = jug2

    def __eq__(self, other):
        return self.jug1 == other.jug1 and self.jug2 == other.jug2

    def __hash__(self):
        return hash((self.jug1, self.jug2))

def dfs(current_state, visited, jug1_capacity, jug2_capacity, target_volume):
    if current_state.jug1 == target_volume:
        print("Jug 1 now has", target_volume, "liters.")
        return True

    visited.add(current_state)

    # Define all possible operations: (action, from_jug, to_jug)
    operations = [
        ('Fill Jug 1', jug1_capacity, current_state.jug2),
        ('Fill Jug 2', current_state.jug1, jug2_capacity),
        ('Empty Jug 1', 0, current_state.jug2),
        ('Empty Jug 2', current_state.jug1, 0),
        ('Pour Jug 1 to Jug 2',
            max(0, current_state.jug1 + current_state.jug2 - jug2_capacity),
            min(jug2_capacity, current_state.jug1 + current_state.jug2)),
        ('Pour Jug 2 to Jug 1',
            min(jug1_capacity, current_state.jug1 + current_state.jug2),
            max(0, current_state.jug1 + current_state.jug2 - jug1_capacity))
    ]

    for operation in operations:
        action, new_jug1, new_jug2 = operation
        new_state = WaterJugState(new_jug1, new_jug2)

        if new_state not in visited:
            print(f"Trying: {action} => ({new_jug1}, {new_jug2})")
            if dfs(new_state, visited, jug1_capacity, jug2_capacity, target_volume):
                return True
    return False

def solve_water_jug_problem(jug1_capacity, jug2_capacity, target_volume):
    initial_state = WaterJugState(0, 0)
    visited = set()

    if dfs(initial_state, visited, jug1_capacity, jug2_capacity, target_volume):
        print("Solution found!")
    else:
        print("Solution not possible.")

# Example usage:
def main():
    jug1_capacity = int(input("Enter Jug 1 capacity: "))
```

```
    jug2_capacity = int(input("Enter Jug 2 capacity: "))
    target_volume = int(input("Enter Target Volume for Jug 1: "))

    if target_volume > jug1_capacity:
        print("The target volume is greater than the capacity of Jug 1. No solution possible.")
        return

    print(f"Solving Water Jug Problem with capacities ({jug1_capacity}, {jug2_capacity}) to measure
{target_volume} liters in Jug 1.")
    solve_water_jug_problem(jug1_capacity, jug2_capacity, target_volume)

if __name__ == "__main__":
    main()
```

```
------------------------------ OUTPUT----------------------------------------------------------------
Enter Jug 1 capacity: 3
Enter Jug 2 capacity: 2
Enter Target Volume for Jug 1: 1
Solving Water Jug Problem with capacities (3, 2) to measure 1 liters in Jug 1.
Trying: Fill Jug 1 => (3, 0)
Trying: Fill Jug 2 => (3, 2)
Trying: Empty Jug 1 => (0, 2)
Trying: Pour Jug 2 to Jug 1 => (2, 0)
Trying: Fill Jug 2 => (2, 2)
Trying: Pour Jug 2 to Jug 1 => (3, 1)
Trying: Empty Jug 1 => (0, 1)
Trying: Pour Jug 2 to Jug 1 => (1, 0)
Jug 1 now has 1 liters.
Solution found!
```

Program 2: Implement and Demonstrate Best First Search Algorithm on Missionaries-Cannibals Problems using Python

```
# Python program to illustrate Missionaries & Cannibals Problem

print("\n")
print("\tGame Start\nNow the task is to move all of them to the right side of the river")
print('''rules:
1. The boat can carry at most two people
2. If cannibals number greater than missionaries, the cannibals would eat the missionaries
3. The boat cannot cross the river by itself with no people on board
''')

# Initial counts of Missionaries and Cannibals on the left and right side
lM = 3          # Left side Missionaries number
lC = 3          # Left side Cannibals number
rM = 0          # Right side Missionaries number
rC = 0          # Right side Cannibals number
k = 0           # To count the number of moves

# Initial state of the game
print("\nM M M C C C |     --- | \n")

try:
    while True:
        # Left side -> Right side river travel
        print("Left side -> Right side river travel")

        while True:
            uM = int(input("Enter number of Missionaries to travel => "))
            uC = int(input("Enter number of Cannibals to travel => "))

            # Check if the total number of people in the boat exceeds 2
            if uM + uC > 2:
                print("The boat can carry at most two people. Re-enter:")
            # Check if there are enough missionaries and cannibals on the left side
            elif uM < 0 or uC < 0:
                print("Negative numbers are not allowed. Re-enter:")
            elif (lM - uM) >= 0 and (lC - uC) >= 0:
                break  # Valid input, exit loop
            else:
                print("Not enough people on the left side. Re-enter:")

        # Update the numbers on the left and right side after the move
        lM -= uM
        lC -= uC
        rM += uM
        rC += uC

        # Display the current state
        print("\nCurrent State:")
        print("Left side:", "M " * lM, "C " * lC, "| --> |", "Right side:", "M " * rM, "C " * rC)

        k += 1  # Increment the number of moves
```

```
        # Check if the cannibals have eaten the missionaries
        if (lC > lM and lM > 0) or (rC > rM and rM > 0):
            print("Cannibals eat missionaries:\nYou lost the game")
            break

        # Check if the game is won (all are moved to the right side)
        if (rM + rC) == 6:
            print("You won the game! Congrats")
            print(f"Total attempts: {k}")
            break

        # Right side -> Left side river travel
        print("\nRight side -> Left side river travel")

        while True:
            userM = int(input("Enter number of Missionaries to travel back => "))
            userC = int(input("Enter number of Cannibals to travel back => "))

            # Check if the total number of people in the boat exceeds 2
            if userM + userC > 2:
                print("The boat can carry at most two people. Re-enter:")
            # Check if there are enough missionaries and cannibals on the right side
            elif userM < 0 or userC < 0:
                print("Negative numbers are not allowed. Re-enter:")
            elif (rM - userM) >= 0 and (rC - userC) >= 0:
                break  # Valid input, exit loop
            else:
                print("Not enough people on the right side. Re-enter:")

        # Update the numbers after the move
        lM += userM
        lC += userC
        rM -= userM
        rC -= userC

        # Display the current state
        print("\nCurrent State:")
        print("Left side:", "M " * lM, "C " * lC, "| <-- |", "Right side:", "M " * rM, "C " * rC)

        # Check if the cannibals have eaten the missionaries
        if (lC > lM and lM > 0) or (rC > rM and rM > 0):
            print("Cannibals eat missionaries:\nYou lost the game")
            break

except EOFError as e:
    print("\nInvalid input, please retry!")
```

----------OUTPUT----------

```
        Game Start
Now the task is to move all of them to the right side of the river
```

rules:
1. The boat can carry at most two people
2. If cannibals number greater than missionaries, the cannibals would eat the missionaries
3. The boat cannot cross the river by itself with no people on board

M M M C C C | --- |

Left side -> Right side river travel
Enter number of Missionaries to travel => 1
Enter number of Cannibals to travel => 1

Current State:
Left side: M M C C | --> | Right side: M C

Right side -> Left side river travel
Enter number of Missionaries to travel back => 1
Enter number of Cannibals to travel back => 0

Current State:
Left side: M M M C C | <-- | Right side: C
Left side -> Right side river travel
Enter number of Missionaries to travel => 0
Enter number of Cannibals to travel => 2

Current State:
Left side: M M M | --> | Right side: C C C

Right side -> Left side river travel
Enter number of Missionaries to travel back => 0
Enter number of Cannibals to travel back => 1

Current State:
Left side: M M M C | <-- | Right side: C C
Left side -> Right side river travel
Enter number of Missionaries to travel => 2
Enter number of Cannibals to travel => 0

Current State:
Left side: M C | --> | Right side: M M C C

Right side -> Left side river travel
Enter number of Missionaries to travel back => 1
Enter number of Cannibals to travel back => 1

Current State:
Left side: M M C C | <-- | Right side: M C
Left side -> Right side river travel
Enter number of Missionaries to travel => 2
Enter number of Cannibals to travel => 0

Current State:

```
Left side:  C C  | --> | Right side: M M M  C

Right side -> Left side river travel
Enter number of Missionaries to travel back => 0
Enter number of Cannibals to travel back => 1

Current State:
Left side:  C C C  | <-- | Right side: M M M
Left side -> Right side river travel
Enter number of Missionaries to travel => 0
Enter number of Cannibals to travel => 2

Current State:
Left side:  C  | --> | Right side: M M M  C C

Right side -> Left side river travel
Enter number of Missionaries to travel back => 0
Enter number of Cannibals to travel back => 1

Current State:
Left side:  C C  | <-- | Right side: M M M  C
Left side -> Right side river travel
Enter number of Missionaries to travel => 0
Enter number of Cannibals to travel => 2

Current State:
Left side:   | --> | Right side: M M M  C C C
You won the game! Congrats
Total attempts: 6

=== Code Execution Successful ===
```