Name	**Hatim Yusuf Sawai**
UID no.	**2021300108**
Experiment No.	**1**

AIM:	**Implement an Intelligent agent (problem formulation and implementation) -Water Jug Problem, Missionary Cannibal problem**
PROBLEM DEFINITION:	**1) Missionaries and Cannibal Problem:** In the missionaries and cannibals problem, three missionaries and three cannibals must cross a river using a boat which can carry at most two people, under the constraint that, for both banks, if there are missionaries present on the bank, they cannot be outnumbered by cannibals (if they were, the cannibals would eat the missionaries). The boat cannot cross the river by itself with no people on board. **2) Water Jug Problem:** "You are given two jugs, a 4-liter one and a 3-liter one. Neither has any measuring markers on it. There is a pump that can be used to fill the jugs with water. How can you get exactly 2 liters of water in either of jug."
THEORY:	**1. Missionaries and Cannibals Problem:** The Missionaries and Cannibals problem is a classic river-crossing puzzle that involves three missionaries and three cannibals who must cross a river. The problem statement includes the following constraints: 1. At any side of the river, the number of cannibals should not outnumber the number of missionaries, or the cannibals will eat the missionaries. 2. A small boat is available and can carry at most two individuals at a time.

Solution Approach:

One common approach to solving this problem is using a state-space search algorithm, such as Breadth-First Search (BFS) or Depth-First Search (DFS). Here's a simplified example of how to solve it using BFS:

Start with an initial state where all three missionaries and three cannibals are on one side of the river, and the boat is also on the same side.
Generate possible valid actions, such as moving two people from one side to the other or moving one person from one side to the other.
Apply these actions to create new states while checking that they satisfy the constraints.
Keep track of visited states to avoid revisiting them.
Continue generating and exploring states until you find a state where all individuals have safely crossed to the other side.

2. Water Jug Problem:

The Water Jug problem involves two jugs with different capacities. The goal is to measure out a specific quantity of water using these jugs.

The problem statement includes the following constraints:
1. Jug A has a capacity of x liters, and jug B has a capacity of y liters (where x and y are positive integers).
2. You have an unlimited supply of water.

Solution Approach:

One way to solve the Water Jug problem is through a systematic search for valid states. Here's a simplified example of how to solve it:

Start with both jugs empty.
Define a target quantity of water (z liters) that you want to measure out.
Consider three possible actions: fill a jug, empty a jug, or pour water

	from one jug to another. Apply these actions to create new states and keep track of them. Continue generating and exploring states while ensuring that you don't exceed the jug capacities. Eventually, you will find a state where one of the jugs contains the desired z liters of water, indicating a solution.

CODE:	**1. Missionary & Cannibals Program:**
	`def main():`
	`  print("Welcome to Missionary and Cannibals Game!\nRules:\nNo. of Missionaries cannot be less than no. of cannibals at any point on any bank!\n\nLet's Begin!")`
	`  lm = int(input("Enter the no. of missionaries: "))`
	`  lc = int(input("Enter the no. of cannibals: "))`
	`  b = rm = rc = 0`
	`  print(f"Initial State(on left):\nMissionaries: {lm}\nCannibals: {lc}\nBoat is on left Bank!\n")`
	`  while(True):`
	`  if lm == 0 and lc == 0:`
	`  print("You Won the Game!\n")`
	`  print(f"Final State on Left:\nMissionaries: {lm}\nCannibals: {lc}\n")`
	`  print(f"Final State on Right:\nMissionaries: {rm}\nCannibals: {rc}\n")`
	`  break`
	`  while(True):`
	`  um, uc = map(int, input("Enter the no. of missionaries and cannibals: ").split())`
	`  if um > 2 or uc > 2 or um < 0 or uc < 0 or um + uc > 2:`
	`  print("Invalid Input! Try Again!")`
	`  elif b == 0 and (uc > lc or um > lm):`
	`  print("Invalid Input! Try Again!")`
	`  elif b == 1 and (uc > rc or um > rm):`
	`  print("Invalid Input! Try Again!")`
	`  else:`
	`  break`
	`  if b == 0:`
	`  lm -= um`
	`  lc -= uc`
	`  rm += um`
	`  rc += uc`
	`  else:`

```
            lm += um
            lc += uc
            rm -= um
            rc -= uc
        if (rm < rc and rm != 0) or (lm < lc and lm!= 0):
            print("Missionaries are less than cannibals!\nYou
Lost the Game!")
            break
        else:
            print(f"Current State on Left Bank:\nMissionaries:
{lm}\nCannibals: {lc}\n")
            print(f"Current State on Right Bank:\nMissionaries:
{rm}\nCannibals: {rc}\n")
            if b == 0:
                b = 1
            elif b == 1:
                b = 0
            print("Boat is on left Bank!\n") if b == 0 else
print("Boat is on right Bank!\n")

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

2. Water Jug Program:

```
def main():
    print("Welcome to Water Jug Problem!\nRules:You are given m
and n litre jugs and you have to take out z litres of water to
solve the problem. Jugs can be fully emptied or filled, or
trasnferred from 1 to another!\n\nLet's Begin!!!")
    final = list(map(int, input("Enter quantity of jugs 1 and
2: ").split()))
    z = int(input("Enter Goal Amt: "))
    jugs = list(final)
    while(True):
        print(f"\nCurrent Status:\nJug1: {jugs[0]}L\tJug2:
{jugs[1]}L\n")
        i = int(input("Pick a Jug (1 or 2): "))-1
        choice = int(input("Pick an Action:\n1. Empty\t2.
Fill\n3. Pour to other Jug\n"))
        if choice == 1 and jugs[i] != 0:
            jugs[i] = 0
        elif choice == 2 and jugs[i] != final[i]:
```

```
            jugs[i] = final[i]
        elif choice == 3:
            source = i
            target = 1 - source
            temp = final[target]-jugs[target]
            amt = min(temp, jugs[source])
            jugs[source] -= amt
            jugs[target] += amt
        elif choice == 4:
            continue

        if jugs[0] > final[0] or jugs[1] > final[1]:
            print("A jug has had an overflow! Please start
again!")
            break
        elif jugs[0]==z or jugs[1]==z:
            print(f"\nGoal state achieved!!!\nJug1:
{jugs[0]}L\tJug2: {jugs[1]}L")
            break

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

OUTPUT:

1. Missionary & Cannibals Output:

```
xcadat …\AIML-Practicals on main (✓) via (venv)
→ python -u "d:\SEM_5\AIML-Practicals\Exp1\mcb.py"
Welcome to Missionary and Cannibals Game!
Rules:
No. of Missionaries cannot be less than no. of cannibals at any point on any bank!

Let's Begin!
Enter the no. of missionaries: 3
Enter the no. of cannibals: 3
Initial State(on left):
Missionaries: 3
Cannibals: 3
Boat is on left Bank!

Enter the no. of missionaries and cannibals: 0 2
Current State on Left Bank:
Missionaries: 3
Cannibals: 1

Current State on Right Bank:
Missionaries: 0
Cannibals: 2

Boat is on right Bank!
```

```
Enter the no. of missionaries and cannibals: 0 2
Current State on Left Bank:
Missionaries: 3
Cannibals: 1

Current State on Right Bank:
Missionaries: 0
Cannibals: 2

Boat is on right Bank!

Enter the no. of missionaries and cannibals: 0 1
Current State on Left Bank:
Missionaries: 3
Cannibals: 2

Current State on Right Bank:
Missionaries: 0
Cannibals: 1

Boat is on left Bank!

Enter the no. of missionaries and cannibals: 0 2
Current State on Left Bank:
Missionaries: 3
Cannibals: 0

Current State on Right Bank:
Missionaries: 0
Cannibals: 3

Boat is on right Bank!
```

```
Current State on Left Bank:
Missionaries: 0
Cannibals: 2

Current State on Right Bank:
Missionaries: 3
Cannibals: 1

Boat is on left Bank!

Enter the no. of missionaries and cannibals: 0 2
Current State on Left Bank:
Missionaries: 0
Cannibals: 0

Current State on Right Bank:
Missionaries: 3
Cannibals: 3

Boat is on right Bank!

You Won the Game!

Final State on Left:
Missionaries: 0
Cannibals: 0

Final State on Right:
Missionaries: 3
Cannibals: 3

 xcadat …\AIML-Practicals on  main (✓) via  (venv) took 19m59s
```

2. Water Jug Output:

```
xcadat …\AIML-Practicals on main (✓) via (venv)
→ python -u "d:\SEM_5\AIML-Practicals\Exp1\wj.py"
Welcome to Water Jug Problem!
Rules:You are given m and n litre jugs and you have to take
 or filled, or trasnferred from 1 to another!

Let's Begin!!!
Enter quantity of jugs 1 and 2: 3 4
Enter Goal Amt: 2

Current Status:
Jug1: 3L        Jug2: 4L

Pick a Jug (1 or 2): 2
Pick an Action:
1. Empty        2. Fill
3. Pour to other Jug
1

Current Status:
Jug1: 3L        Jug2: 0L

Pick a Jug (1 or 2): 1
Pick an Action:
1. Empty        2. Fill
3. Pour to other Jug
3

Current Status:
Jug1: 0L        Jug2: 3L

Pick a Jug (1 or 2): 1
Pick an Action:
1. Empty        2. Fill
3. Pour to other Jug
2

Current Status:
Jug1: 3L        Jug2: 3L

Pick a Jug (1 or 2): 1
Pick an Action:
1. Empty        2. Fill
3. Pour to other Jug
3

Goal state achieved!!!
Jug1: 2L        Jug2: 4L
xcadat …\AIML-Practicals on main (✓) via (venv) took 2m51s
→
```

CONCLUSION:	In this experiment, we learned how to make an interactive game to show a solving approach to the given problem definitions.