Q1. Write down the pseudo code for jug problem showing various possibilities for solving it.

Ans.

There are mainly 8 Rules or cases needed to solve the water jug problem which are as follows:

1. Fill the 4 gallon jug completely
2. Fill the 3 gallon jug completely
3. Empty the 4 gallon jug
4. Empty the 3 gallon jug
5. Pour some water from the 3 gallon jug to fill the 4 gallon jug
6. Pour some water from the 4 gallon jug to fill the 3 gallon jug.
7. Pour all water from 3 gallon jug to the 4 gallon jug
8. Pour all water from the 4 gallon jug to the 3 gallon jug

Using these cases the 4 gallon jug can be filled with 2 gallon water in the following ways:

1.

State of the jugs	Case to apply
(0, 0)	2
(0, 3)	7
(3, 0)	2
(3, 3)	5
(4, 2)	3
(0, 2)	7
(2, 0)	Required State

2.

State of the jugs	Case to apply
(0, 0)	1
(4, 0)	6
(1, 3)	4
(1, 0)	7
(0, 1)	1
(4, 1)	6
(2, 3)	4
(2, 0)	Required State

Q2. Discuss constraint satisfaction problem with an algorithm for solving a Crypto Arithmetic problem.

Ans.

A Constraint Satisfaction Problem (CSP) is a mathematical problem that involves finding a solution that satisfies a set of constraints or conditions. In other words, it is a problem where we need to find a feasible solution that satisfies a set of limitations or restrictions.

One way to solve CSPs is using an algorithm called Backtracking. The backtracking algorithm is a depth-first search algorithm that tries to find a feasible solution by testing values for variables one by one. The algorithm backtracks and tries a different value for a variable if the current combination of values leads to an infeasible solution.

An example of a CSP is a Crypto Arithmetic problem, where we have to find values for a set of letters that satisfy a set of arithmetic constraints. For example, consider the problem of finding values for the letters S, E, N, D, M, O, R, Y such that the following equation holds:

SEND + MORE = MONEY

We can use the backtracking algorithm to solve this problem by assigning values to the letters one by one, starting with the leftmost letter. If we reach a point where the current combination of values leads to an infeasible solution, we backtrack and try a different value for the previous letter.

The algorithm continues until we find a feasible solution or exhaust all possible combinations of values. In this case, we will eventually find the solution S = 9, E = 5, N = 6, D = 7, M = 1, O = 0, R = 8, Y = 2.

An algorithm for solving a Crypto Arithmetic problem would involve the following steps:

1. Create a list of variables, one for each letter in the expression.
2. Define the constraints, which specify the relationships between the variables and the values they can take.
3. Assign initial values to the variables, and evaluate the constraints.
4. If the constraints are not satisfied, adjust the values of the variables until a solution is found.
5. Continue adjusting the values of the variables until all constraints are satisfied.

This algorithm can be implemented using search algorithms, such as backtracking or depth-first search, which allow for systematic exploration of the solution space until a solution is found. The algorithm can also be implemented using constraint programming, which allows for a more declarative and intuitive representation of the problem and constraints.

Q3. List down the characteristics of intelligent agent.

Ans.

In AI, an intelligent agent can be defined as a system that perceives its environment and takes actions to maximize its chances of success in a given task or goal. The following are the characteristics of an intelligent agent:

1. Autonomy: The agent should be capable of operating without human intervention.
2. Reactivity: The agent should be able to respond to changes in the environment.
3. Proactivity: The agent should be able to initiate actions on its own, without waiting for explicit instructions.
4. Learning: The agent should be able to improve its performance over time based on experience.
5. Adaptability: The agent should be able to adapt to new situations and changing environments.
6. Goal-directed behavior: The agent should have a goal or set of goals that it is trying to achieve.

Q4. Identify the problems encountered during hill climbing and list the ways available to deal with these problems.

Ans.

1. Local Maximum: A local maximum is a peak state in the landscape which is better than each of its neighboring states, but there is another state also present which is higher than the local maximum.

Solution: Backtracking technique can be a solution of the local maximum in state space landscape. Create a list of the promising path so that the algorithm can backtrack the search space and explore other paths as well.

2. Plateau: A plateau is the flat area of the search space in which all the neighbor states of the current state contains the same value, because of this algorithm does not find any best direction to move. A hill-climbing search might be lost in the plateau area.

Solution: The solution for the plateau is to take big steps or very little steps while searching, to solve the problem. Randomly select a state which is far away from the current state so it is possible that the algorithm could find non-plateau region.

3. Ridges: A ridge is a special form of the local maximum. It has an area which is higher than its surrounding areas, but itself has a slope, and cannot be reached in a single move.

Solution: With the use of bidirectional search, or by moving in different directions, we can improve this problem.

Q5. Solve the Water Jug problem.

Ans.

Following are the rules or cases for required to solve the Water Jug problem:

S.No.	Initial State	Condition	Final state	Description of action taken
1.	(x,y)	If x<4	(4,y)	Fill the 4 gallon jug completely
2.	(x,y)	if y<3	(x,3)	Fill the 3 gallon jug completely
3.	(x,y)	If x>0	(0,y)	Empty the 4 gallon jug
4.	(x,y)	If y>0	(x,0)	Empty the 3 gallon jug
5.	(x,y)	If (x+y)<7	(4, y-[4-x])	Pour some water from the 3 gallon jug to fill the 4 gallon jug
6.	(x,y)	If (x+y)<7	(x-[3-y],y)	Pour some water from the 4 gallon jug to fill the 3 gallon jug.
7.	(x,y)	If (x+y)<4	(x+y,0)	Pour all water from 3 gallon jug to the 4 gallon jug
8.	(x,y)	if (x+y)<3	(0, x+y)	Pour all water from the 4 gallon jug to the 3 gallon jug

```
def solve(case_value, x, y):
    switch = {
        '1':case1,
        '2':case2,
        '3':case3,
        '4':case4,
        '5':case5,
        '6':case6,
        '7':case7,
        '8':case8
    }
    func = switch.get(case_value, lambda: "Invalid case")
    return func(x, y)


def case1(x, y):
    x = 4
```

```
    return x, y


def case2(x, y):
    y = 3
    return x, y


def case3(x, y):
    x = 0
    return x, y


def case4(x, y):
    y = 0
    return x, y


def case5(x, y):
    while (x < 4):
        x += 1
        y -= 1
    return x, y


def case6(x, y):
    while (y < 3):
        x -= 1
        y += 1
    return x, y


def case7(x, y):
```

```
    if x + y > 4:
        x = 4
        y = 0
    else:
        x += y
        y = 0
    return x, y

def case8(x, y):
    if x + y > 3:
        x = 0
        y = 3
    else:
        y += x
        x = 0
    return x, y



x = y = 0
print(f"4G jug = {x}, 3G jug = {y}\n")
while True:
    take_input = str(input("Enter Case to apply: "))
    x, y = solve(take_input, x, y)
    print(f"4G jug = {x}, 3G jug = {y}\n")
    if x == 2 and y == 0:
        print("Good Job!!!")
        break
```

Solution:

Let 4 gallon jug be X and 3 gallon be Y;

Starting from initial State

X, Y = 0, 0

Filling Y completely

X, Y = 0, 3

Pouring all water from Y into X

X, Y = 3, 0

Filling Y completely again

X, Y = 3, 3

Pouring water from Y into X till X is full

X, Y = 4, 2

Emptying X

X, Y = 0, 2

Pouring all water from Y into X

X, Y = 2, 0

Required State is achieved.