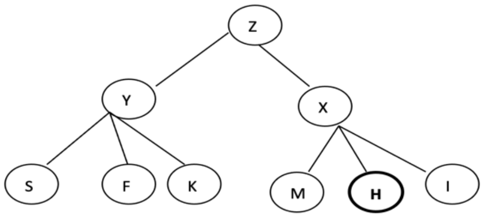
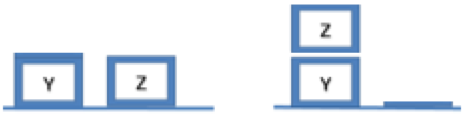
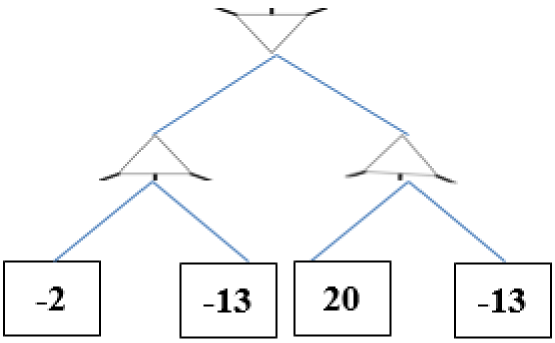
SASTRA DEEMED TO BE UNIVERSITY (U/S 3 OF THE UGC ACT, 1956) THINK MERIT | THINK TRANSPARENCY | THINK SASTRA	**School of Computing** **Third CIA Exam – May 2025** Course Code: INT314 Course Name: Artificial Intelligence and Logical Reasoning Duration: 90 minutes Max Marks: 50

Answer ANY FOUR questions PART A 8 x 5 = 40 Marks

Q.No	Questions	CO	RBT Level
1	You're analyzing customer behavior to understand who is likely to buy organic food. You observe three things: Whether a person is health conscious, Whether a person buys organic food, Whether the person uses reusable bags. From the full joint probability distribution table find a. What is the probability that a customer is **health conscious**? (1) b. What is the probability of reusable bags? (1) c. What is the probability that a customer **buys organic given they are health conscious**? (4) d. What is the probability that a customer is **health conscious given they use a reusable bag**? (4) **Health Conscious | Organic | Reusable Bag | Probability** Yes | Yes | Yes | 0.20 Yes | Yes | No | 0.05 Yes | No | Yes | 0.10 Yes | No | No | 0.05 No | Yes | Yes | 0.05 No | Yes | No | 0.05 No | No | Yes | 0.15 No | No | No | 0.35	4	Apply
2	Illustrate Instance and ISA Relationships of FOL. (10)	3	Understand
3	A bot is searching a spare part in the available rooms. The part is available in the bold-rounded room. Apply Depth First Search and	1	

	analyze the searching process in a table using the order of fringe queue. (10)  		
4	a) Recall Minimax algorithm. (7) b) Recall hill climbing algorithm. (3)	2 2	Rememb er
5	a) Construct goal stack planning for the given block world example. (10)  Initial State Goal State	4	Apply

Part B **1x 10 = 10 Marks**

Q.No	**Questions**	**CO**	**RBT Level**
6	a) Recall the steps of resolution refutation proof of FOL. (5) b) Apply Alpha beta pruning to the given tree, find alpha, beta values, root value and pruned branches. (2)  c) Define Modus Ponens. (3)	3 2 3	Unde rstan d Apply Reme mber

Answers:

1. What is the probability that a customer is health conscious?

We sum all rows where **Health Conscious = Yes**:

- (Yes, Yes, Yes): 0.20
- (Yes, Yes, No): 0.05
- (Yes, No, Yes): 0.10
- (Yes, No, No): 0.05

P(Health Conscious)=0.20+0.05+0.10+0.05=0.40

What is the probability of Reusable Bag?

P(Reusable Bag)

- (Yes, Yes, Yes): 0.20
- (Yes, No, Yes): 0.10
- (No, Yes, Yes): 0.05
- (No, No, Yes): 0.15

P(Reusable Bag)=0.20+0.10+0.05+0.15=0.50

What is the probability that a customer buys organic given they are health conscious?

We apply conditional probability:

P(Organic∣Health Conscious)=P(Organic∧Health Conscious) / P(Health Conscious)

From part (a):

P(Health Conscious)=0.40

Rows where both are true:

- (Yes, Yes, Yes): 0.20
- (Yes, Yes, No): 0.05
  P(Organic∧Health Conscious)=0.20+0.05=0.25
  So:
  P(Organic∣Health Conscious)=0.25/0.40=0.625

What is the probability that a customer is health conscious given they use a reusable bag?

P(Health Conscious∣Reusable Bag)=P(Health Conscious∧Reusable Bag)/P(Reusable Bag)

Rows with **Health Conscious = Yes and Reusable Bag = Yes**:

- (Yes, Yes, Yes): 0.20
- (Yes, No, Yes): 0.10

So:

P(Health Conscious∧Reusable Bag)=0.20+0.10=0.30

P(Reusable Bag)=0.50

P(Health Conscious∣Reusable Bag)=0.30/0.50=0.60

2. Illustrate Instance and ISA Relationships of FOL.

- Specific attributes instance and isa play an important role particularly in a useful form of reasoning called property inheritance.

- The predicates instance and isa explicitly captured the relationships they used to express, namely class membership and class inclusion.

(first 5), class membership represented with unary predicates (such as Roman), each of which corresponds to a class.

- Asserting that P(x) is true is equivalent to asserting that x is an instance (or element) of P.

1. Man(Marcus).
2. Pompeian(Marcus).
3. $\forall x$: Pompeian(x) $\rightarrow$ Roman(x).
4. ruler(Caesar).
5. $\forall x$: Roman(x) $\rightarrow$ loyalto(x, Caesar) $\vee$ hate(x, Caesar).

The second part of the figure contains representations that use the instance predicate explicitly.

- The predicate instance is a binary one, whose first argument is an object and whose second argument is a class to which the object belongs.

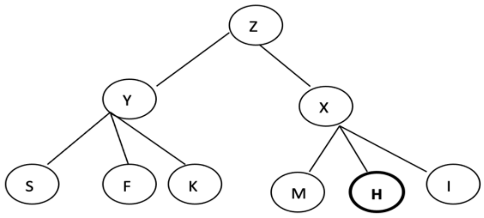
- But these representations do not use an explicit isa predicate.
- Instead, subclass relationships, such as that between Pompeians and Romans, described as shown in sentence 3.
- The implication rule states that if an object is an instance of the subclass Pompeian then it is an instance of the superclass Roman.
- Note that this rule is equivalent to the standard set-theoretic definition of the subclass- superclass relationship.

1. instance(Marcus, man).
2. instance(Marcus, Pompeian).
3. $\forall x$: instance(x, Pompeian) $\rightarrow$ instance(x, Roman).
4. instance(Caesar, ruler).
5. $\forall x$: instance(x, Roman). $\rightarrow$ loyalto(x, Caesar) $\vee$ hate(x, Caesar).

- The third part contains representations that use both the instance and isa predicates explicitly.
- • The use of the isa predicate simplifies the representation of sentence 3, but it requires that one additional axiom (shown here as number 6) be provided.

1. instance(Marcus, man).
2. instance(Marcus, Pompeian).
3. isa(Pompeian, Roman)
4. instance(Caesar, ruler).
5. $\forall x$: instance(x, Roman). $\rightarrow$ loyalto(x, Caesar) $\vee$ hate(x, Caesar).
6. $\forall x$: $\forall y$: $\forall z$: instance(x, y) $\wedge$ isa(y, z)$\rightarrow$ instance(x, z).

3. A bot is searching a spare part in the available rooms. The part is available in the bold-rounded room. Apply Depth First Search and analyze the searching process in a table using the order of fringe queue. (10)



Ans:LIFO Queue

Marks- (4) (2) (2) (1) (1)

Iteration	Fringe	Closed list (Visited)	Goal test	Removed nodes from memory	Back track to
0	Z		Z ×		
1	X,Y	Z	Y×		
2	X,K,F,S	Z,Y	S×	S	Y
3	X,K,F	Z,Y	F×	F	Y
4	X,K	Z,Y	K×	K	Y
5	X	Z		Y	Z
6	I,H,M	Z,X	M×	M	X
7	I,H	Z,X	H- Goal reached ✓		

4a)

```
function MINIMAX-DECISION(state) returns an action
  inputs: state, current state in game

  v ← MAX-VALUE(state)
  return the action in SUCCESSORS(state) with value v
```

```
function MAX-VALUE(state) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← −∞
  for a, s in SUCCESSORS(state) do
    v ← MAX(v, MIN-VALUE(s))
  return v
```

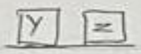
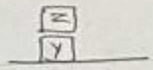
```
function MIN-VALUE(state) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← ∞
  for a, s in SUCCESSORS(state) do
    v ← MIN(v, MAX-VALUE(s))
  return v
```

b)

```
function HILL-CLIMBING(problem) returns a state that is a local maximum
  inputs: problem, a problem
  local variables: current, a node
                   neighbor, a node

  current ← MAKE-NODE(INITIAL-STATE[problem])
  loop do
    neighbor ← a highest-valued successor of current
    if VALUE[neighbor] ≤ VALUE[current] then return STATE[current]
    current ← neighbor
```

5 a) Goal Stack Planning

Initial — Goal

ontable(Y) ∧ ontable(Z) ∧ Clear(Y) ∧ Clear(Z)

ontable(Y) ∧ on(Z, Y) ∧ clear(Z)

Start from Goal push the predicate into stack.

clear(Z)
on(Z, Y)
on(Z, Y) ∧ clear(Z)

Goal state

~~Clear(Z)~~
on(Z, Y)

Remove the predicates which are true

On(Z, Y) is not true according to the current/Initial state. So push the actions to achieve the goal of on(Z, Y)

Stack(Z, Y)
On(Z, Y)

Now push the pre conditions of Stack(Z, Y)

Remove the pre conditions which are true →

~~clear(Y)~~ → P
Holding(Z) → P
Stack(Z, Y)
On(Z, Y)

Holding Z is not true, So add actions to achieve it

~~Holding(Z)~~
Stack(Z, Y)
On(Z, Y)

pickup(Z)
~~onTable(Z)~~
Stack(Z, Y)
On(Z, Y)

add pre-conditions for pickup(Z)

Clear(Z) → P
Armempty → P
} true, ∴ pop them

Pickup(Z)
Stack(Z, Y)
On(Z, Y)

~~Pickup(Z)~~
Stack(Z, Y)
On(Z, Y)

Plan = {pickup(Z)} → apply action at current world

plan = {pickup(Z), stack(Z, Y)}

~~Stack(Z, Y)~~
On(Z, Y)

~~On(Z, Y)~~

On(Z, Y) is true

pop it. ∴ reached goal.

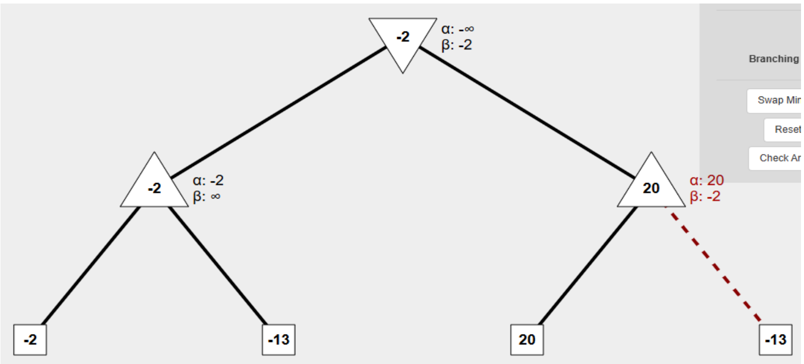
Answer the question **PART B** **1x 10 = 10 Marks**

1. a) Recall the steps of resolution refutation proof of FOL. (5)

Resolution refutation proofs involve the following steps:

1. Put the premises or axioms into *clause form* .
2. Add the negation of what is to be proved, in clause form, to the set of axioms.
3. *Resolve* these clauses together, producing new clauses that logically follow from them.
4. Produce a contradiction by generating the empty clause.
5. The substitutions used to produce the empty clause are those under which the opposite of the negated goal is true.

b) Apply Alpha beta pruning to the given tree, find alpha, beta values, root value and pruned branches. (2)



c) Define Modus Ponens. (3)

This section covers standard patterns of inference that can be applied to derive chains of conclusions that lead to the desired goal. These patterns of inference are called **inference rules.** The best-known rule is called **Modus Ponens** and is written as follows:

$$\frac{\alpha \Rightarrow \beta, \quad \alpha}{\beta}$$

The notation means that, whenever any sentences of the form a $\Rightarrow \beta$ and a are given, then the sentence β can be inferred. For example, if (*WumpusAhead A WumpusAlive)* $\Rightarrow$ *Shoot* and (*WumpusAhead A WumpusAlive)* are given, then *Shoot* can be inferred.

Another useful inference rule is **And-Elimination,** which says that, from a conjunction, any of the conjuncts can be inferred:

$$\frac{\alpha \wedge \beta}{a}$$

For example, from (*WumpusAhead A WumpusAlive), WumpusAlive* can be inferred.

By considering the possible truth values of a and β, one can show easily that Modus Ponens and And-Elimination are sound once and for all. These rules can then be used in any particular instances where they apply, generating sound inferences without the need for enumerating models.