USN | | | | | | | | | | |

Internal Assessment Test 2 – May 2025

Sub:	**ARTIFICIAL INTELLIGENCE**					Sub Code:	BAD402	Branch:	**AInDS**
Date:	**26/05/2025**	Duration:	**90 minutes**	Max Marks:	**50**	Sem	**IV**		**OBE**

		Answer any FIVE Questions	**MARKS**	**CO**	**RBT**
1		**Explain the A* search to minimize the total estimated cost** **A* search explanation: 5 Marks** **Total estimated cost: 5 Marks** A* (A-star) search is an informed search algorithm designed to find the most efficient path to a goal while minimizing the total estimated cost. It is widely used in pathfinding and graph traversal due to its ability to guarantee the shortest path to the goal in most cases. A* achieves this by combining the actual cost to reach a node with a heuristic estimate of the remaining cost to reach the goal, helping it to focus on the most promising paths. Key Concepts of A* Search A* search operates on the principle of minimizing the evaluation function f(n) for each node n, where: f(n)=g(n)+h(n) Here: • g(n) is the cost of the path from the starting node to the current node n. • h(n) is the heuristic estimate of the cost from n to the goal node. How A* Minimizes the Total Estimated Cost 1. Total Cost Minimization: A* calculates the total cost f(n)f(n)f(n) for each node, combining both the actual cost so far g(n)g(n)g(n) and the estimated cost to the goal h(n)h(n)h(n). By prioritizing nodes with the lowest f(n)f(n)f(n) values, A* balances both the actual path cost and the estimated remaining cost. 2. Optimality with Admissible Heuristics: When the heuristic function h(n)h(n)h(n) is admissible (never overestimates the actual cost to reach the goal), A* is guaranteed to find the shortest path to the goal. Admissibility ensures that h(n)h(n)h(n) provides an optimistic estimate, guiding the search toward efficient paths without underestimating the true cost. 3. Consistency (Monotonicity): If the heuristic is also consistent (meaning h(n)⩽c(n,m)+h(m)h(n) \leq c(n, m) + h(m)h(n)⩽c(n,m)+h(m), where c(n,m)c(n, m)c(n,m) is the cost from node nnn to mmm), A* will avoid re-exploring nodes, making it even more efficient.	[10]	3	L3

2		**Write an algorithm for hill climbing search and explain in detail.** **Explanation: 2 Marks** **Types: 4 Marks** **Algorithm: 4 Marks** Hill Climbing is an optimization search algorithm used to find a solution that maximizes or minimizes a particular objective function by iteratively improving the current state. It's often used in scenarios where you want to find a local maximum (or minimum) in the solution space. The idea behind hill climbing is simple: start from an initial state, then move in the direction that best improves the objective until no further improvements can be made. Hill climbing is a greedy algorithm that always seeks to make moves that immediately increase (or decrease) the objective function value. However, because it only evaluates neighboring states, it's prone to getting stuck in local optima rather than the global optimum. Types of Hill Climbing 1. Simple Hill Climbing: Moves only to neighboring states that improve the objective function; stops when no improvement is possible. 2. Steepest-Ascent Hill Climbing: Evaluates all neighbors and chooses the one that maximizes the improvement in the objective function. 3. Stochastic Hill Climbing: Chooses randomly among neighbors that improve the objective function, adding randomness to avoid local optima. 4. Random-Restart Hill Climbing: Runs multiple hill climbing processes from different random starting points to increase the likelihood of finding the global optimum. Algorithm for Hill Climbing Search The basic Hill Climbing algorithm is as follows: 1. Initialize: Start from an initial state. 2. Loop: 1. Generate Neighboring States: Create a set of all possible states reachable from the current state. 2. Evaluate: For each neighbor, calculate its objective function value. 3. Move to the Best Neighbor: ▪ If there is a neighbor that has a better objective function value than the current state, move to that neighbor. ▪ If no neighbor improves the objective function, terminate the algorithm. 3. Return the current state as the best solution found.	[10]	3	L3

3		**Explain the following with respect to first order logic** **i) Assertions and queries in first order logic ii) The kinship domain iii) Numbers, sets, and lists**	[10]	2	L2

First order logic definition: 1 Mark
Each Types: 3 Marks , Three types so total 9 Marks

i) Assertions and Queries in First-Order Logic

Assertions:

- Assertions in FOL are statements that declare facts about objects in the domain of discourse. These facts can be expressed using predicates, constants, and logical connectives.
- For example, an assertion might state that "Alice is a student" can be represented as:

$$P(\text{Alice})$$

where $P(x)$ is a predicate that denotes "x is a student."

- Assertions can also use quantifiers:
  - **Universal Assertion:** $\forall x P(x)$ states that "for all x, x is a student."
  - **Existential Assertion:** $\exists y Q(y)$ states that "there exists a y such that y is mortal."

Queries:

- Queries are expressions used to inquire about the existence or properties of objects in the domain. They are often used in knowledge representation systems to retrieve information based on the existing assertions.
- For example, a query could be represented as:

$$\exists y(R(\text{Alice}, y))$$

which asks, "Does there exist a y such that Alice loves y?"

- Another query might be:

$$\forall x(P(x) \rightarrow Q(x))$$

asking whether "for every x, if x is a student, then x is mortal."

ii) The Kinship Domain

The kinship domain is a classic example in logic that deals with relationships among family members. It uses first-order logic to represent various familial relations such as parent, child, sibling, etc.

Predicates in the Kinship Domain:

- Common predicates might include:
  - $\text{Parent}(x, y)$: "x is a parent of y."
  - $\text{Child}(x, y)$: "x is a child of y."
  - $\text{Sibling}(x, y)$: "x is a sibling of y."
  - $\text{Grandparent}(x, y)$: "x is a grandparent of y."
  - $\text{Cousin}(x, y)$: "x is a cousin of y."

Example Assertions:

- To assert that Alice is Bob's parent, we write:

$$\text{Parent}(\text{Alice}, \text{Bob})$$

- To state that Bob has a sibling, we could use:

$$\exists y \text{Sibling}(\text{Bob}, y)$$

Rules in the Kinship Domain:

- Rules can be defined to express relationships:
  - A person x is a grandparent of y if x is a parent of z and z is a parent of y:

$$\text{Grandparent}(x, y) \leftrightarrow \exists z(\text{Parent}(x, z) \wedge \text{Parent}(z, y))$$

iii) Numbers, Sets, and Lists in First-Order Logic

Numbers:

- In first-order logic, natural numbers can be represented using predicates and functions. For example, the successor function $S(n)$ can be used to denote the next number after n. Thus, one can express properties about numbers, such as:
  - $\text{Even}(n)$: "n is even," which can be defined in terms of the successor function.
- To express arithmetic relationships, axioms can be introduced:
  - For example, to assert that the sum of two numbers is also a number:

$$\forall x \forall y \exists z(z = x + y)$$

4		**Explain the syntax and semantics of the first order logic** **<u>Syntax: 5 Marks</u>** **<u>Semantics: 5 Marks</u>** First-order logic (FOL), also known as predicate logic or first-order predicate calculus, extends propositional logic by introducing quantifiers and predicates, allowing for more expressive statements about objects and their relationships. It provides a formal framework for reasoning about the properties of objects and their interrelations. Syntax of First-Order Logic The syntax of first-order logic includes several key components: 1. Constants: o Constants are symbols that represent specific objects in the domain of discourse. For example, a,b,c can be constants representing specific individuals. 2. Variables: o Variables (e.g., x,y,z) are symbols that can represent any object in the domain. They are often used in quantification. 3. Predicates: o Predicates are functions that represent properties or relations among objects. A predicate can take one or more arguments. For example: ▪ P(x)could represent a property (e.g., "x is a person"). ▪ R(x,y) could represent a relation (e.g., "x loves y"). 4. Functions: o Functions map objects from the domain to other objects. For example, a function f(x)might represent "the parent of x." 5. Logical Connectives: o The same logical connectives from propositional logic are used: ▪ Negation (¬) ▪ Conjunction (∧) ▪ Disjunction (∨) ▪ Implication (→) ▪ Biconditional (↔) 6. Quantifiers: o First-order logic introduces quantifiers to express statements about all or some objects in the domain:	[10]	2	L2

Q. No.		Question	Marks	CO	RBT
5		**Explain marginalization and normalization with a full joint distribution of (toothache, catch, cavity)** **<u>Marginalization: 5 Marks</u>** **<u>Normalization: 5 Marks</u>**	[10]	1	L2

* To understand marginalization and normalization within the context of a full joint distribution of the random variables

Joint probability Distribution

* A full joint distribution provides the probability of every possible combination of the random variables, for simplicity, lets assume each variable is binary : Toothache (T), catch (C), and Cavity (Cav)
* The joint probability distribution might look like this

Toothache	Catch	Cavity	P(Toothache, Catch, Cavity)
T	T	T	0.108
T	T	F	0.012
T	F	T	0.072
T	F	F	0.008
F	T	T	0.016
F	T	F	0.064
F	F	T	0.144
F	F	F	0.576

Marginalization

* it is the process of summing the probabilities over one or more variables to obtain the marginal probability distribution of the remaining variables.

ex: to find the marginal probability P(cavity) we sum over the toothache and catch variables

Normalization

Normalization ensures that the probabilities in a distribution sum to 1

* if we have a probability distribution that doesn't sum to 1, we can normalize it by dividing each probability by the total sum of all probabilities
* lets say we have the following unnormalized distribution

Cavity	P(Cavity)
T	0.34
F	0.66

in this case the sum is 1, so normalization is not required. However, suppose our probabilities were not correctly normalized and summed to 0.85 instead

$\Sigma P(\text{cavity}) = 0.85$

$P(\text{Cavity} = T) = \frac{0.34}{0.85} \approx 0.4$

$P(\text{Cavity} = F) = \frac{0.66}{0.85} \approx 0.78$

After normalization :

6		**write the representation of Bayes Theorem. In a class, 70% children were fall sick due to Viral fever and 30% due to Bacterial fever. The probability of observing temperature for Viral is 0.78 and for Bacterial is 0.31. If a child develops high temperature, find the child's probability of having viral infection.** **<u>Formula: 2 Marks</u>** **<u>Derivation: 6 Marks</u>** **<u>Answer: 2 Marks</u>** $\Rightarrow P(A/B) = \frac{P(B/A) \cdot P(A)}{P(B)}$ given $P(V) = 0.70$ $P(B) = 0.30$ $P(T/V) = 0.78$ $P(T/B) = 0.31$ $P(V/T) = ?$ $P(T) = ?$ $\quad P(V/T) = \frac{P(T/V) \cdot P(V)}{P(T)}$ $P(T) = P(T/V) \cdot P(V) + P(T/B) \cdot P(B)$ $= (0.78 \times 0.70) + (0.31 \times 0.30)$ $P(T) = 0.639$ $P(V/T) = \frac{0.78 \times 0.70}{0.639}$ $P(V/T) = 0.854$	[10]	2	L3

CI **CCI** **HOD**