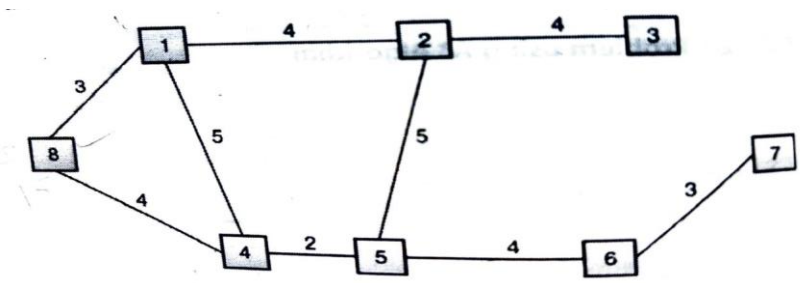
| (Affiliated to University of Mumbai) | ***End Semester Examination (R-24/R-25) SH 2026 Answer Key with marking scheme*** |
|---|---|
| Branch: AIML | Course: Artificial Intelligence |
| Year/ Semester: TE/ V | Course code: AIMLC501 |
| Time: 03 hours | Marks: 80 |

| | | Marks |
|---|---|---|
| Q. 1 | All questions are compulsory. | |
| A. | Describe any two real-world case studies of AI in healthcare or robotics.<br>• **Healthcare – Google's diabetic retinopathy screening:** Deep learning models were trained on retinal images to detect diabetic retinopathy with accuracy comparable to ophthalmologists. Deployed in clinics in India and Thailand, it enabled early detection in areas lacking specialist eye care.<br>• **Robotics – Amazon warehouse robots (Kiva/Drive units):** Autonomous mobile robots navigate warehouses using sensors and AI-based path planning to carry shelves to human pickers, dramatically improving order-fulfillment speed and reducing worker walking time. | 05 |
| B. | Explain Artificial Intelligence. Explain the Turing Test with a suitable example.<br><br>**Artificial Intelligence**<br>AI is the branch of computer science concerned with building systems that perform tasks normally requiring human intelligence—such as learning, reasoning, perception, language understanding, and decision-making. It ranges from rule-based systems to machine learning and deep neural networks that improve through data and experience.<br>**Turing Test with example**<br>Proposed by Alan Turing (1950), the test judges a machine's intelligence by conversation: a human interrogator chats via text with both a machine and a hidden human. If the interrogator cannot reliably tell which is which, the machine is said to pass. Example: if you text with two unseen entities and cannot decide which one is the computer, the computer has passed the Turing Test. | 05 |
| C. | Describe with example Partial order planning.<br>**Partial Order Planning with example**<br>Partial order planning builds plans by adding actions and ordering constraints only where necessary, leaving independent actions unordered (hence "partial order") rather than committing to a fixed sequence. Example: to "buy groceries and clean the house," the planner records that shopping and cleaning must both happen but imposes no order between them—only constraints like "go to store before buying items." This flexibility avoids unnecessary commitments and reduces replanning. | 05 |
| D. | Describe the Min-Max Algorithm with an Example.<br><br>Min-Max is a recursive decision rule for two-player zero-sum games: MAX chooses moves maximizing the score, MIN chooses moves minimizing it, alternating up the game tree. Example: in tic-tac-toe, from a position MAX | 05 |

| | | |
|---|---|---|
| | evaluates all moves; for each, MIN's best reply is assumed, and MAX picks the move whose worst-case outcome is best (e.g., choosing a move guaranteeing a draw over one risking a loss). | |
| E. | E. Reward, policy and value in Reinforcement Learning (5 marks)<br>• Reward: immediate scalar feedback received after an action/state transition; defines the objective. (2)<br>• Policy π: rule/probability distribution specifying which action an agent takes in a state. (1.5)<br>• Value: expected cumulative future discounted reward from a state (Vπ(s)) or state-action pair (Qπ(s,a)) under a policy. (1.5) | 05 |
| F. | Illustrate with examples different quantifiers.<br><br>**Quantifiers with examples**<br>• **Universal quantifier (∀, "for all"):** ∀x (Human(x) → Mortal(x)) — "All humans are mortal."<br>• **Existential quantifier (∃, "there exists"):** ∃x (Human(x) ∧ Wise(x)) — "Some human is wise."<br>• **Nested example:** ∀x ∃y Loves(x, y) — "Everyone loves someone" (different from ∃y ∀x Loves(x, y), "someone is loved by everyone"). | 05 |
| | | |
| Q.2A | Explain Hill Climbing Algorithm and discuss the problems that occurs in hill climbing algorithm. Explain how those limitations can be resolved.<br><br>**Hill Climbing Algorithm**<br>Hill climbing is a local search optimization technique that starts with an initial state and repeatedly moves to a neighboring state that improves the objective function (higher value for maximization, lower for minimization). It accepts only improvements and stops when no neighbor is better—reaching a **local optimum**. A variant is **steepest ascent**, which evaluates all neighbors and picks the best. It is simple and memory-efficient (keeps only the current state) but is not complete or optimal.<br>**Problems in Hill Climbing**<br>1. **Local maxima (or minima):** The algorithm gets stuck at a peak that is better than its neighbors but not the global best.<br>2. **Plateaus:** Flat regions where all neighbors have equal value, so no move improves—the agent wanders or stalls, unsure of direction.<br>3. **Ridges:** Narrow elongated peaks where moving in any single direction decreases value, though moving diagonally would improve—single-step moves fail to climb.<br>4. **Foothills/Shoulders:** A shoulder is a plateau followed by a rise; lack of gradient information can cause the search to stop prematurely.<br>**How limitations are resolved**<br>• **Random restart:** Run hill climbing from multiple random initial states and keep the best result—helps escape local maxima.<br>• **Simulated annealing:** Occasionally accept worse moves with a probability that decreases over time, allowing escape from local optima.<br>• **Tabu search / memory:** Keep a list of recently visited states to avoid revisiting and cycling, and force exploration of new regions.<br>• **Stochastic/randomized hill climbing:** Choose a random improving move (or random sideways moves) to handle plateaus and ridges. | |

| | | |
|---|---|---|
| | • **Beam search / multiple agents:** Maintain several candidate states in parallel (local beam search) to explore diverse regions.<br>• **Better heuristics / larger neighborhoods:** Redesign the evaluation function or allow multi-step (diagonal) moves to handle ridges and shoulders. | |
| | OR | |
| Q.2B. | Consider the graph given in the fig. Assume that the initial state is S and the goal state is 7. Find a path from initial state to goal state using A* search. Also report the solution cost. The straight line distance heuristic estimates for the nodes are as follows:<br>h(1) =14,h(2)=10,h(3)=8,h(4)=12,h(5)=10,h(6)=10,h(S)=15.<br><br><br>**Given Data:**<br>• **Heuristics:** h(S)=15, h(1)=14, h(2)=10, h(3)=8, h(4)=12, h(5)=10, h(6)=10, h(7)=0.<br>• **Edge Costs:** S-1:3, S-4:4, 1-2:4, 1-4:5, 2-3:4, 2-5:5, 4-5:2, 5-6:4, 6-7:3.<br>**A* Search Evaluation: f(n) = g(n) + h(n)**<br>Where g(n) is the cost from S to node n, and h(n) is the heuristic estimate from n to the goal.<br>**Step 1:** Initialize the open list with the start node **S(8)**.<br>• Node: **S** \| g=0 \| h=15 \| **f = 0 + 15 = 15**<br>• Open List: [S(15)]<br>**Step 2:** Expand **S(8)**. Neighbors are 1 and 4.<br>• Path to **1**: g=3, h=14 → f = 3 + 14 = **17**<br>• Path to **4**: g=4, h=12 → f = 4 + 12 = **16**<br>• Open List: [4(16), 1(17)]<br>**Step 3:** Expand node **4** (lowest f=16). Neighbors are 1 and 5.<br>• Path to **1**: g = 4+5=9, h=14 → f = 9+14 = 23 (Worse than existing 17, so ignore)<br>• Path to **5**: g = 4+2=6, h=10 → f = 6+10 = **16**<br>• Open List: [5(16), 1(17)]<br>**Step 4:** Expand node **5** (lowest f=16). Neighbors are 2 and 6.<br>• Path to **2**: g = 6+5=11, h=10 → f = 11+10 = **21**<br>• Path to **6**: g = 6+4=10, h=10 → f = 10+10 = **20**<br>• Open List: [1(17), 6(20), 2(21)]<br>**Step 5:** Expand node **1** (lowest f=17). Neighbors are 2 and 4 (S is visited).<br>• Path to **2**: g = 3+4=7, h=10 → f = 7+10 = **17** (Better than existing 21, so update 2's parent to 1) | |

| | | |
|---|---|---|
| | • Path to **4**: g = 3+5=8, h=12 → f = 8+12 = 20 (Worse than existing 16, so ignore)<br>• Open List: [2(17), 6(20)]<br>**Step 6:** Expand node **2** (lowest f=17). Neighbors are 3 and 5 (1 is visited).<br>• Path to **3**: g = 7+4=11, h=8 → f = 11+8 = **19**<br>• Path to **5**: g = 7+5=12, h=10 → f = 12+10 = 22 (Worse than existing 16, so ignore)<br>• Open List: [3(19), 6(20)]<br>**Step 7:** Expand node **3** (lowest f=19). Neighbor is 2 (visited). Dead end.<br>• Open List: [6(20)]<br>**Step 8:** Expand node **6** (lowest f=20). Neighbor is 7.<br>• Path to **7**: g = 10+3=13, h=0 → f = 13+0 = **13**<br>• Open List: [7(13)]<br>**Step 9:** Expand node **7** (Goal). The goal test succeeds. Stop the search.<br>**Final Solution**<br>Tracing back the optimal path from goal to start using the updated parents:<br>7 ← 6 ← 5 ← 4 ← S (8)<br>• **Path: S → 4 → 5 → 6 → 7**<br>• **Solution Cost: 13** *(Calculated as: S-4(4) + 4-5(2) + 5-6(4) + 6-7(3) = 13)*<br>• | |
| | | |
| Q.3A | Introduce Reinforcement Learning. Explain the concepts of reward, policy, value function, and the difference between model-based and model-free approaches.<br><br>**Reinforcement Learning and Key Concepts**<br>**Reinforcement Learning (RL):**<br>RL is a machine learning paradigm where an **agent** learns to make decisions by interacting with an **environment**. Instead of being given explicit correct answers, the agent learns through trial and error by taking actions and receiving feedback in the form of rewards or penalties, aiming to maximize its total reward over time.<br>**Key Concepts:**<br>• **Reward:** A scalar feedback signal received after taking an action. It defines the goal of the RL problem (e.g., +10 points for winning a game, -1 for crashing a robot).<br>• **Policy ($\pi$):** The agent's strategy. It is a mapping from states to actions (i.e., what action to take in a given situation). The goal of RL is to find the optimal policy.<br>• **Value Function:** A prediction of the total future reward an agent can expect to accumulate from a given state (or state-action pair). It evaluates how "good" it is to be in a particular state.<br>**Model-Based vs. Model-Free Approaches:**<br>• **Model-Based:** The agent builds or is given a predictive model of the environment's dynamics (how actions change states and yield rewards). It uses this model to plan ahead before acting (e.g., playing chess by simulating future moves).<br>• **Model-Free:** The agent learns directly by trial and error without knowing or building a model of the environment. It learns the policy or value function purely through experience (e.g., Q-learning or teaching a robot to walk through repetition). | |

| | | |
|---|---|---|
| | | |
| | OR | |
| Q.3B | Tabular reinforcement learning Definition with example-5M Dynamic programming and Poilcy Iteration with suitable example-5M | |
| | | |
| Q.4A | Explain the steps involved in converting the propositional logic statement into CNF.<br>Consider the following Axioms.<br>Rani is hungry.<br>If rani is hungry she barks.<br>If rani is barking then raja is angry.<br>1) Represent these axioms in FOL.<br>2) Convert the FOL to CNF.<br><br>3) Prove that Raja is angry by using resolution technique<br><br>**1. Steps to Convert to CNF**<br>1. Eliminate biconditionals (↔).<br>2. Eliminate implications (→) using $P \to Q \equiv \neg P \vee Q$.<br>3. Move negations inward using De Morgan's laws ($\neg\neg P \equiv P$).<br>4. Standardize variables (rename them uniquely).<br>5. Skolemize (remove existential quantifiers ∃).<br>6. Drop universal quantifiers (∀).<br>7. Distribute OR over AND.<br>8. Extract the final clauses.<br>**2. FOL Representation**<br>• **Axiom 1:** $Hungry(Rani)$<br>• **Axiom 2:** $Hungry(Rani) \to Barks(Rani)$<br>• **Axiom 3:** $Barks(Rani) \to Angry(Raja)$<br>**3. Conversion to CNF**<br>• **C1:** $Hungry(Rani)$<br>• **C2:** $\neg Hungry(Rani) \vee Barks(Rani)$<br>• **C3:** $\neg Barks(Rani) \vee Angry(Raja)$<br>**4. Resolution Proof**<br>To prove $Angry(Raja)$, we add its negation to our clauses:<br>• **C4:** $\neg Angry(Raja)$*(Negated Goal)*<br>**Resolution steps:**<br>1. Resolve **C1** and **C2** → $Barks(Rani)$<br>2. Resolve $Barks(Rani)$and **C3** → $Angry(Raja)$<br>3. Resolve $Angry(Raja)$and **C4** →**Empty Clause** { }(Contradiction)<br>Since we reached a contradiction, the negated goal is false. **Therefore, Raja is angry is proven true.** | |
| | OR | |
| Q.4B | Forward chaining (10 marks)<br>Rules: (Croaks(x)∧EatsFlies(x))→Frog(x); (Chirps(x)∧Sings(x))→Canary(x); Frog(x)→Green(x); Canary(x)→Yellow(x). Fact: Croaks(Fritz) and EatsFlies(Fritz). (2)<br>Match both antecedents of the frog rule with the facts and infer Frog(Fritz). (3) | |

| | | |
|---|---|---|
| | Apply Frog(x)→Green(x) with x=Fritz to infer Green(Fritz). (3)<br>Thus Y = green. The canary rule is not triggered because no facts establish that Fritz chirps and sings. (2) | |
| | | |
| Q.5A | Describe Hidden Markov Models (HMMs). Explain the Forward-Backward algorithm and Viterbi algorithm briefly.<br><br>**1. Hidden Markov Models (HMMs), Forward-Backward, and Viterbi**<br>**Hidden Markov Models (HMMs):**<br>An HMM is a statistical model used to represent sequences where the system being modeled is assumed to be a Markov process with unobservable (hidden) states. It consists of:<br>• **Hidden States:** Internal states that cannot be directly observed (e.g., "Sunny" or "Rainy").<br>• **Observations:** The visible outputs that depend on the hidden states (e.g., whether someone carries an umbrella).<br>• **Parameters:** Initial state probabilities, state transition probabilities (moving from one hidden state to another), and emission probabilities (probability of an observation given a hidden state).<br>**Forward-Backward Algorithm:**<br>This algorithm computes the probability of being in a specific hidden state at a given time, given a sequence of observations.<br>• **Forward pass:** Calculates the probability of ending up in a state at time $t$given all past observations.<br>• **Backward pass:** Calculates the probability of seeing all future observations given the state at time $t$.<br>• Combining both gives the smoothed probability of a state at time $t$, which is crucial for tasks like speech recognition to figure out the exact state at a specific moment.<br>**Viterbi Algorithm:**<br>Used for finding the **most likely sequence of hidden states** (the optimal path) that resulted in a given sequence of observations. It uses dynamic programming: at each time step, it keeps track of the most probable path leading to each state. It is widely used in bioinformatics (e.g., gene finding) and natural language processing (e.g., part-of-speech tagging). | |
| | OR | |
| Q.5B | Explain Bayesian Networks in detail. Describe how inference is performed in Bayesian Networks with a suitable example<br><br>**Bayesian Networks and Inference**<br>**Bayesian Networks (BNs):**<br>A Bayesian Network is a Directed Acyclic Graph (DAG) that represents a set of random variables and their conditional dependencies.<br>• **Nodes:** Represent random variables (e.g., "Rain", "Traffic").<br>• **Edges (Arrows):** Represent direct causal or probabilistic dependencies (e.g., Rain →Traffic). | |

- **Conditional Probability Tables (CPTs):** Each node has a CPT that quantifies the effect of its parent nodes.
  They are highly efficient for representing joint probability distributions compactly, making them ideal for handling uncertainty.

**Inference in Bayesian Networks:**

Inference is the process of computing the posterior probability of a set of query variables given some observed evidence.

- **Example Setup:**
  - **Node A (Rain):** P(Rain) = 0.2
  - **Node B (Traffic):** Depends on Rain. P(Traffic | Rain) = 0.9; P(Traffic | ¬Rain) = 0.3.
- **Inference Task:** Suppose you observe heavy traffic (Evidence = Traffic). You want to find the probability that it is raining (Query = Rain). We need to compute $P(\text{Rain} \mid \text{Traffic})$.
- **Calculation:** Using Bayes' Theorem:

P(\text{Rain} \mid \text{Traffic}) = \frac{P(\text{Traffic} \mid \text{Rain}) \cdot P(\text{Rain})}{P(\text{Traffic})}
]

First, find the total probability of Traffic:

$$P(\text{Traffic}) = P(\text{Traffic} \mid \text{Rain})P(\text{Rain}) + P(\text{Traffic} \mid \neg\text{Rain})P(\neg\text{Rain})$$
$$P(\text{Traffic}) = (0.9 \times 0.2) + (0.3 \times 0.8) = 0.18 + 0.24 = 0.42$$

Now, calculate the posterior:

$$P(\text{Rain} \mid \text{Traffic}) = \frac{0.9 \times 0.2}{0.42} = \frac{0.18}{0.42} \approx 0.428 \ (42.8\%)$$

- **Methods:** For complex networks, exact inference is done using algorithms like **Variable Elimination** or **Belief Propagation** (for trees). For very large networks, **Approximate Inference** methods like Markov Chain Monte Carlo (MCMC) or particle filtering are used

| | | |
|---|---|---|
| | | |
| Q.6A | Ethical and regulatory considerations in AI healthcare and robotics (10 marks)<br>Privacy/security: protect sensitive data, minimize collection, control access and secure storage/transmission. (2)<br>Safety/validation: rigorous testing, clinical/operational validation, risk management, fail-safe design and post-deployment monitoring. (2)<br>Fairness: assess demographic/data bias and performance across populations; mitigate unequal outcomes. (2)<br>Transparency/accountability: document limitations, provide understandable explanations where feasible, define human responsibility and auditability. (2)<br>Consent, compliance and human oversight: obtain appropriate consent, follow applicable data-protection/medical-device rules, enable clinician/operator override, and address liability, cybersecurity and responsible use in robotics. (2) | |
| OR | | |
| Q.6B | Examine the applications of AI in medical imaging, disease diagnosis, and healthcare prediction. | |

| | | |
|---|---|---|
| | • **Medical Imaging:** AI (particularly deep learning/CNNs) is widely used to analyze X-rays, MRIs, and CT scans. It aids in detecting tumors, fractures, or anomalies (e.g., detecting diabetic retinopathy from retinal scans or lung nodules from CTs) often with accuracy comparable to human radiologists, speeding up triage and diagnosis.<br>• **Disease Diagnosis:** AI assists in diagnosing complex diseases by analyzing diverse data. For example, IBM Watson for Oncology suggests cancer treatments by parsing medical literature, and AI models analyze genetic data (genomics) to diagnose rare genetic disorders or predict disease susceptibility.<br>• **Healthcare Prediction (Predictive Analytics):** AI uses Electronic Health Records (EHRs) to predict patient outcomes. Examples include predicting the likelihood of a patient developing sepsis, forecasting hospital readmission rates, and identifying high-risk patients for chronic diseases (like diabetes or heart failure) so preventive care can be administered early. | |
| *********** All the Best*********** | | |