DEPARTMENT ELECTRONICS & COMMUNICATION ENGINEERING

VTU Exam-Jan-2025 QP Solution

Name of the course :	**Intelligent Systems and Machine Learning**	**Sub Code :**	**BEC515A**
Name of the Faculty/s :	**Sushma B, Richa Tengshe**	**Sem & Sec :**	**5 (A, B, C and D)**

	Question and solution	Marks	
1a	Explain the history of AI. 1. Early Foundations (Before 1950s) ● The idea of **intelligent machines** dates back to ancient myths and automata. ● **Mathematical logic** (e.g., works of Aristotle, Leibniz, and Boole) laid the foundation for reasoning. ● **Alan Turing (1936)** proposed the **Turing Machine**, and later (1950) introduced the **Turing Test** to define machine intelligence. 2. Birth of AI (1950s - 1960s) ● **1956: Dartmouth Conference** – Considered the birth of AI; John McCarthy coined the term "Artificial Intelligence." ● **Early AI programs:** ○ **Logic Theorist (1955)** by Newell & Simon – First AI program to prove mathematical theorems. ○ **General Problem Solver (1957)** – Aimed at solving any problem but had limitations. ● Development of **Perceptron (1958)** by Frank Rosenblatt, an early neural network model. 3. The Rise and Fall of AI (1960s - 1970s) ● AI research flourished with early expert systems and symbolic reasoning. ● However, **lack of computing power** and **limited understanding of learning** led to the first **AI Winter** (1970s), where funding and interest declined. 4. Expert Systems & Knowledge-Based AI (1980s) ● AI regained momentum with **Expert Systems** like **MYCIN (medical diagnosis)** and **XCON (business applications)**. ● **Machine learning concepts** began to develop, but AI was still rule-based.	10	

- Another AI slowdown occurred due to high costs and limitations of rule-based systems.

5. The Rise of Machine Learning & AI Boom (1990s - 2000s)

- AI shifted from rule-based systems to **probabilistic reasoning and data-driven approaches**.
- **Breakthroughs in Machine Learning, Natural Language Processing (NLP), and Robotics** occurred.
- **IBM's Deep Blue (1997)** defeated world chess champion Garry Kasparov.

6. Modern AI & Deep Learning Era (2010s - Present)

- AI exploded with advancements in **Big Data, Deep Learning, and Computing Power**.
- **Key breakthroughs:**
  - **2011:** IBM Watson won Jeopardy!
  - **2012:** AlexNet revolutionized deep learning for image recognition.
  - **2016:** AlphaGo defeated world champion Go player.
  - **2018-Present:** GPT, BERT, and other AI models transformed NLP.
- AI is now widely used in **self-driving cars, medical diagnosis, virtual assistants, and automation**.

7. The Future of AI

- AI research focuses on **explainability, ethics, and Artificial General Intelligence (AGI)**.
- Challenges include **bias, security, and ensuring AI benefits humanity**.

AI approaches are categorized into:

1. **Thinking humanly** (Cognitive modeling)
2. **Thinking rationally** (Logic-based AI)
3. **Acting humanly** (Turing Test approach)
4. **Acting rationally** (Rational agent approach)

1b

Define AI, Explain the foundation of AI?

10

The study of agents that receive percepts from the environment and perform actions."

This definition is based on the concept of **intelligent agents**, which perceive their environment and take actions to maximize their chances of success.

AI is categorized into four main approaches:

1. **Thinking humanly** – Understanding human cognition and replicating it.
2. **Thinking rationally** – Using logic-based systems for reasoning.
3. **Acting humanly** – Passing the **Turing Test** to exhibit human-like behavior.

4. **Acting rationally** – Creating agents that make optimal decisions.

Foundations of AI

The foundations of AI come from multiple disciplines, which have contributed to its development:

1. Philosophy (400 BC – Present)

- Early philosophers like **Aristotle** explored logical reasoning.
- **René Descartes (1637)** and **Leibniz (1700s)** discussed mechanized thinking.
- **Alan Turing (1950)** proposed the **Turing Test**, a key idea in AI.

2. Mathematics (1800s – Present)

- **Boolean Algebra (George Boole, 1854)** – Logical representation of reasoning.
- **Probability Theory (Bayes, 1763)** – Foundation for AI decision-making.
- **Computability Theory (Turing, 1936)** – Theoretical basis for algorithms.

3. Psychology & Cognitive Science (1900s – Present)

- **Cognitive models** (1950s) aimed to simulate human thought processes.
- **Neuroscience** helped in developing artificial neural networks.

4. Linguistics (1950s – Present)

- AI uses **Natural Language Processing (NLP)** to understand and generate human language.
- **Chomsky's Theory of Language (1957)** influenced AI's approach to syntax and semantics.

5. Computer Science & Engineering (1950s – Present)

- Development of **programming languages** like LISP (1958) and Prolog (1970s).
- Evolution of **machine learning algorithms** and **robotics**.

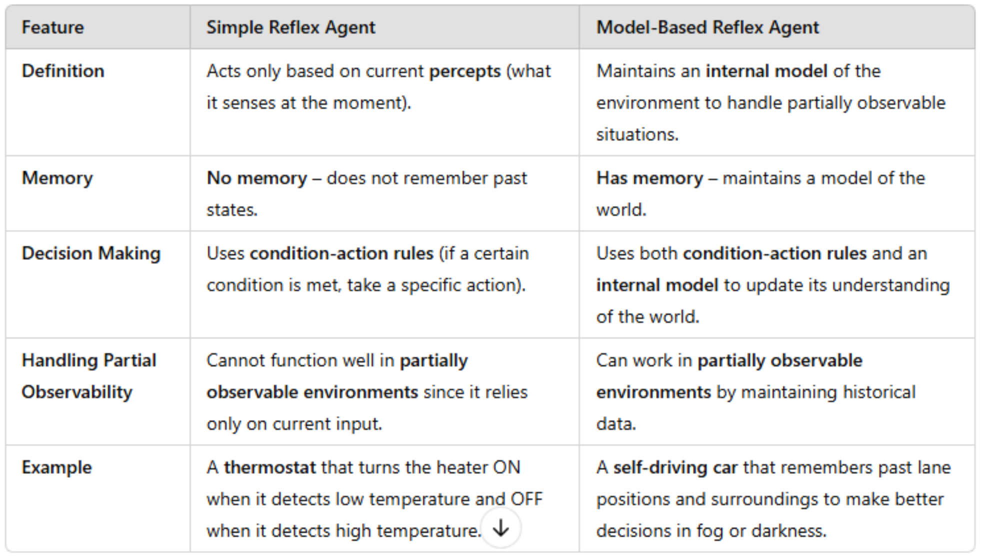
AI is built upon principles from **logic, mathematics, psychology, and computer science**. Russell and Norvig emphasize the **rational agent approach**, where AI is designed to make the best possible decisions in any given situation.

2a	Explain properties of the task environment?	10	

To design an AI system effectively, the following **properties of the task environment** are considered:

i) **Observability**

	● **Fully Observable:** The agent has complete knowledge of the environment at all times (e.g., Chess). ● **Partially Observable:** The agent has limited information about the environment (e.g., Poker, Self-driving cars in fog). ii) **Deterministic vs. Stochastic** ● **Deterministic:** The next state of the environment is fully determined by the agent's actions (e.g., Chess). ● **Stochastic:** The next state is uncertain due to randomness or external factors (e.g., Robot navigation with unpredictable obstacles). iii) **Episodic vs. Sequential** ● **Episodic:** Each decision is independent of past actions (e.g., Image recognition tasks). ● **Sequential:** Past actions affect future decisions (e.g., Chess, Driving). iv) **Static vs. Dynamic** ● **Static:** The environment does not change while the agent is deciding (e.g., Turn-based board games). ● **Dynamic:** The environment changes continuously or while the agent is making a decision (e.g., Real-time video games, Stock market). v) **Discrete vs. Continuous** ● **Discrete:** The environment has a finite number of possible states and actions (e.g., Chess, Tic-Tac-Toe). ● **Continuous:** The environment has an infinite range of possible states and actions (e.g., Self-driving cars, Robotics). vi) **Single-Agent vs. Multi-Agent** ● **Single-Agent:** The agent operates alone with no competitors or allies (e.g., Solving a puzzle). ● **Multi-Agent:** The agent interacts with other agents, which can be cooperative (team-based) or competitive (e.g., Chess, Autonomous vehicle traffic). vii) **Known vs. Unknown** ● **Known:** The agent understands the rules and model of the environment (e.g., Chess, where all rules are predefined). ● **Unknown:** The agent must learn the environment's rules through exploration and experience (e.g., Reinforcement learning in a new video game).		
2b	Differentiate between simple reflex agents and model based reflex agents.	10	



Feature	Simple Reflex Agent	Model-Based Reflex Agent
Definition	Acts only based on current **percepts** (what it senses at the moment).	Maintains an **internal model** of the environment to handle partially observable situations.
Memory	**No memory** – does not remember past states.	**Has memory** – maintains a model of the world.
Decision Making	Uses **condition-action rules** (if a certain condition is met, take a specific action).	Uses both **condition-action rules** and an **internal model** to update its understanding of the world.
Handling Partial Observability	Cannot function well in **partially observable environments** since it relies only on current input.	Can work in **partially observable environments** by maintaining historical data.
Example	A **thermostat** that turns the heater ON when it detects low temperature and OFF when it detects high temperature.	A **self-driving car** that remembers past lane positions and surroundings to make better decisions in fog or darkness.

3a List and explain the components required to define a problem 10

In **Artificial Intelligence (AI)**, defining a problem properly is crucial for designing an **intelligent agent** that can find a solution efficiently.

i) Initial State

- The **starting point** of the problem.
- The agent begins its search from this state.
- Example: In a **chess game**, the initial state is the standard starting position of all pieces.

ii) Actions (Possible Moves)

- A set of **legal operations** that the agent can perform.
- These define how the agent transitions from one state to another.
- Example: In **Tic-Tac-Toe**, an action is placing "X" or "O" in an empty square.

iii) Transition Model

- A function that describes **what happens when an action is taken**.
- It determines the **resulting state** after performing an action.
- Example: In a **maze navigation problem**, if the agent moves "UP" from (x, y), the new state is (x, y+1).

iv) Goal State (Goal Test)

- Defines the **desired solution** or end condition.
- The problem is **solved** when the agent reaches this state.
- Example: In **pathfinding (GPS navigation)**, the goal state is reaching the destination

city.

v) **Path Cost (Cost Function)**

- A numeric value that **measures the efficiency** of a solution.
- Used to **evaluate and compare** different paths to the goal.
- Example: In **Google Maps**, the path cost could be **distance, time, or toll fees**.

Example: Defining the "8-Puzzle" Problem

1. **Initial State:** A shuffled board configuration.
2. **Actions:** Moving the empty tile up, down, left, or right.
3. **Transition Model:** Defines how the board changes after a move.
4. **Goal State:** The tiles are arranged in numerical order.
5. **Path Cost:** The number of moves taken to reach the goal.

3b Explain the goal formulation and problem formulation with examples 10

In AI, **goal formulation** and **problem formulation** are the first steps in designing an intelligent agent to solve a given task.

Goal Formulation

- The **goal** is the **desired outcome** or **end state** that an agent wants to achieve.
- Goals are typically defined based on **the agent's perception of the environment**.
- The goal must be **precise, achievable, and measurable**.

Example: Self-Driving Car

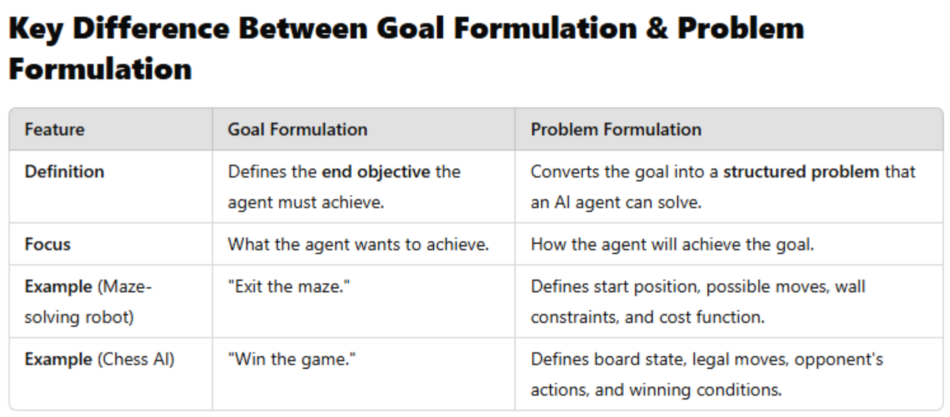
- **Goal:** Reach the destination safely while following traffic rules and optimizing travel time.
- The AI system determines **actions** (speed, lane changes, stops) to achieve this goal.

Problem Formulation

- Once the **goal** is defined, the agent must define a **problem** in a way that it can be solved systematically.
- Problem formulation involves:
    1. **Defining the initial state** (Where is the agent now?)
    2. **Listing possible actions** (What actions can the agent take?)
    3. **Describing the transition model** (What happens when an action is performed?)
    4. **Defining the goal state** (What does success look like?)
    5. **Setting a path cost function** (What is the cost of each action?)

Example: Self-Driving Car (Problem Formulation)

1. **Initial State:** The car is at a starting location (GPS coordinates).
2. **Actions:** Move forward, turn left, turn right, stop, accelerate, etc.
3. **Transition Model:** If the car moves forward at speed X, it reaches a new position Y.
4. **Goal State:** The car reaches the destination safely.
5. **Path Cost:** Minimize time, avoid accidents, and follow traffic rules.



Key Difference Between Goal Formulation & Problem Formulation

Feature	Goal Formulation	Problem Formulation
Definition	Defines the **end objective** the agent must achieve.	Converts the goal into a **structured problem** that an AI agent can solve.
Focus	What the agent wants to achieve.	How the agent will achieve the goal.
Example (Maze-solving robot)	"Exit the maze."	Defines start position, possible moves, wall constraints, and cost function.
Example (Chess AI)	"Win the game."	Defines board state, legal moves, opponent's actions, and winning conditions.

4a Explain breadth first search and depth first search with an example

BFS and DFS are two fundamental **uninformed search algorithms** used in AI to explore and find solutions in state spaces (graphs or trees).

Breadth-First Search (BFS)

Definition:

- **BFS explores all nodes at the current depth level before moving to the next level.**
- It uses a **queue (FIFO - First In, First Out)** data structure.

Algorithm Steps:

1. Start from the **initial node** (root).
2. Explore all **immediate child nodes**.
3. Move to the next level and repeat.
4. Continue until the **goal node** is found or all nodes are explored.

Example: BFS on a Graph

Consider the following graph:

```
    A
   / \
  B   C
 / \   \
D   E   F
```

BFS Traversal Order:

1. Start from **A** → `[A]`
2. Visit its children: **B, C** → `[A, B, C]`
3. Visit **B's** children: **D, E** → `[A, B, C, D, E]`
4. Visit **C's** child: **F** → `[A, B, C, D, E, F]`

Output Order: A → B → C → D → E → F

Pros & Cons of BFS:

Guaranteed to find the shortest path (if all edges have equal weight).
Consumes more memory as it stores all nodes at the current level before moving deeper.

4b Illustrate the different methods of evaluating an algorithm's performance

Evaluating an algorithm's performance is essential to determine its efficiency, scalability, and suitability for a given problem. The following are the key methods used to evaluate an algorithm:

Time Complexity Analysis

- Measures **how the runtime of an algorithm grows** with input size (`n`).
- Expressed using **Big-O Notation** (e.g., **O(n), O(log n), O(n^2)**).
- Helps in comparing algorithms based on their **efficiency in execution time**.

Example: Sorting Algorithms

- **Bubble Sort**: `O(n²)` (slow for large `n`)
- **Merge Sort**: `O(n log n)` (better for large `n`)

Use case: When selecting a sorting or searching algorithm for large datasets.

Completeness

Definition:

- Determines **whether an algorithm is guaranteed to find a solution** if one exists.

Example:

- **BFS: Complete** (always finds a solution if one exists).
- **DFS: Not necessarily complete** (may get stuck in an infinite branch).

Use case: Ensuring an algorithm doesn't miss a valid solution.

Space Complexity Analysis

Definition:

- Evaluates the **amount of memory required** by an algorithm.
- Includes memory for **input, auxiliary variables, recursion stack, and output**.
- Expressed in **Big-O notation** (e.g., `O(n)`, `O(1)`).

Example:

- **BFS:** `O(b^d)` (stores all nodes at the current level).
- **DFS:** `O(d)` (stores only the current path).

Use case: Choosing between search algorithms for memory-limited environments.

Space Complexity Analysis

- Evaluates the **amount of memory required** by an algorithm.
- Includes memory for **input, auxiliary variables, recursion stack, and output**.
- Expressed in **Big-O notation** (e.g., `O(n)`, `O(1)`).

Example: Recursive vs. Iterative Fibonacci

- **Recursive Fibonacci**: `O(n)` space (due to function calls in recursion stack).
- **Iterative Fibonacci**: `O(1)` space (uses only a few variables).

Use case: When designing memory-efficient applications (e.g., embedded systems).

Optimality

Definition:

- Checks whether an algorithm **always finds the best (shortest or least costly) solution**.

Example:

- **BFS (in an unweighted graph): Optimal** (finds the shortest path).
- **DFS: Not optimal** (may find a suboptimal solution first).

Use case: Choosing between algorithms when an optimal solution is required.

Empirical Performance Analysis

Definition:

- Measures real-world execution time by running the algorithm on **actual datasets**.
- Uses metrics like:
  - **CPU Time**
  - **Memory Usage**
  - **Cache Efficiency**

Example:

- Running *A search** vs. **Dijkstra’s Algorithm** on different graph sizes.

Use case: Testing algorithms in **real-time AI applications**.

5a Describe greedy best search as an informed search strategy

Introduction to Informed Search

An **informed search strategy** uses additional knowledge (heuristics) to guide the search toward the goal more efficiently than uninformed methods like BFS or DFS.

Greedy Best-First Search (GBFS) is a heuristic-based search algorithm that selects the next node based on its estimated cost to the goal.

Definition of Greedy Best-First Search

- GBFS **expands the node that appears to be closest to the goal** based on a heuristic function.
- The heuristic function **h(n)** estimates the cost from node `n` to the goal.
- Unlike *A search**, GBFS does **not** consider the cost from the start node, only the estimated distance to the goal.

Evaluation Function:

f(n)=h(n)f(n) = h(n)f(n)=h(n)

where:

- h(n)h(n)h(n) = estimated cost from node `n` to the goal.

Greedy Best-First Search prioritizes low heuristic values.

Algorithm Steps

1. **Initialize an empty priority queue (min-heap).**
2. **Insert the starting node into the queue.**
3. **While the queue is not empty:**
    - **Remove the node with the lowest heuristic value h(n).**
    - **If this node is the goal, return the solution.**
    - **Otherwise, expand its neighbors and add them to the queue.**
4. **Repeat until a solution is found or the search space is exhausted.**

Example: Pathfinding in a Graph

Consider the following graph, where numbers represent h(n) values (estimated cost to goal `G`):

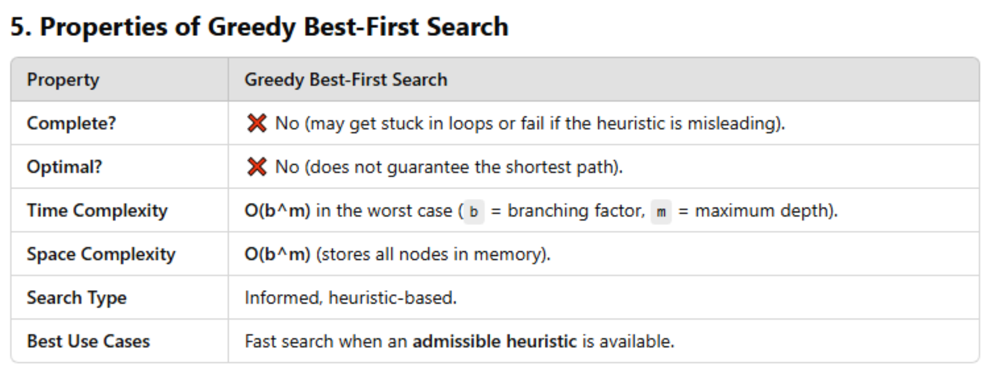
```
      A (10)

  /  \

 B(5)   C(4)

 |  \    |

D(7) E(2) F(0) [Goal]
```

GBFS Traversal (Starting from A, Goal = F):

1. **Start from A (h=10) → Expand B (h=5) and C (h=4).**
2. **Choose C (h=4) (smallest h-value).**
3. **Expand F (h=0) → Goal reached!**

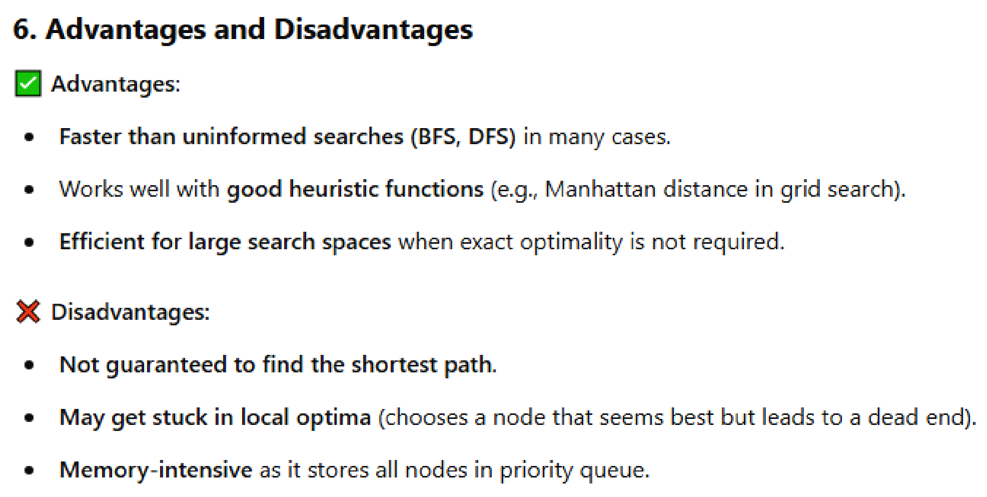
Path found: `A → C → F`

Issue: It does not guarantee the shortest path (e.g., `A → B → E → F` **might be better).**



5. Properties of Greedy Best-First Search

Property	Greedy Best-First Search
Complete?	❌ No (may get stuck in loops or fail if the heuristic is misleading).
Optimal?	❌ No (does not guarantee the shortest path).
Time Complexity	O(b^m) in the worst case (`b` = branching factor, `m` = maximum depth).
Space Complexity	O(b^m) (stores all nodes in memory).
Search Type	Informed, heuristic-based.
Best Use Cases	Fast search when an **admissible heuristic** is available.



6. Advantages and Disadvantages

✅ Advantages:

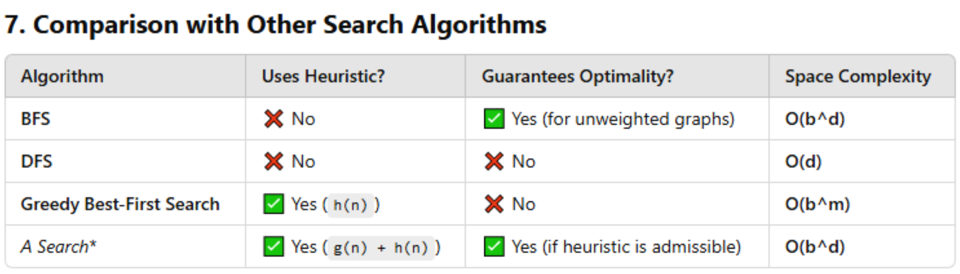
- **Faster than uninformed searches (BFS, DFS)** in many cases.
- Works well with **good heuristic functions** (e.g., Manhattan distance in grid search).
- **Efficient for large search spaces** when exact optimality is not required.

❌ Disadvantages:

- **Not guaranteed to find the shortest path.**
- **May get stuck in local optima** (chooses a node that seems best but leads to a dead end).
- **Memory-intensive** as it stores all nodes in priority queue.

Applications of Greedy Best-First Search

- **Pathfinding algorithms (Google Maps, GPS navigation).**
- **AI in games (NPC movement, shortest enemy paths).**
- **Robotics (navigation and obstacle avoidance).**
- **Machine Learning (feature selection in optimization problems).**



7. Comparison with Other Search Algorithms

Algorithm	Uses Heuristic?	Guarantees Optimality?	Space Complexity
BFS	❌ No	✅ Yes (for unweighted graphs)	O(b^d)
DFS	❌ No	❌ No	O(d)
Greedy Best-First Search	✅ Yes (`h(n)`)	❌ No	O(b^m)
*A Search**	✅ Yes (`g(n) + h(n)`)	✅ Yes (if heuristic is admissible)	O(b^d)

5b Explain knowledge based agent with a generic knowledge based agent program

A **Knowledge-Based Agent (KBA)** is an intelligent agent that:

- Uses **knowledge representation** to model the world.
- Uses **inference mechanisms** to make decisions.
- Can **update its knowledge base** dynamically.

KBAs work by perceiving the environment, reasoning based on stored knowledge, and taking actions accordingly.

Components of a Knowledge-Based Agent

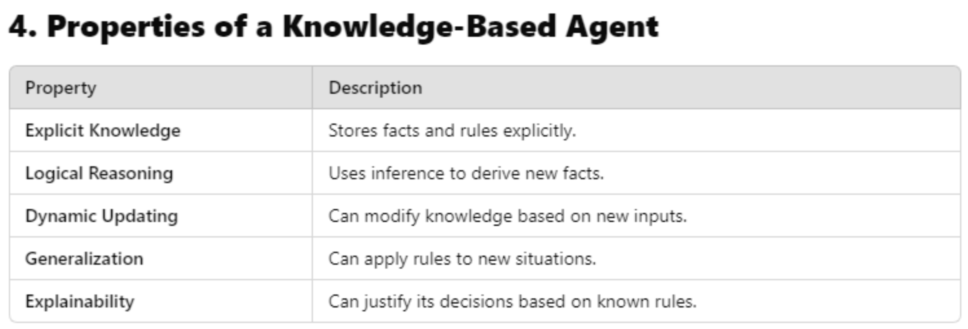
A KBA consists of the following:

1. **Knowledge Base (KB)**
    - **Stores facts, rules, and heuristics about the world.**
    - **Expressed using First-Order Logic (FOL), Propositional Logic, or semantic networks.**
2. **Inference Engine**
    - **Uses logical deduction (e.g., Modus Ponens) to infer new facts.**
    - **Example: If "It is raining" → "Carry an umbrella."**
3. **Perception Module**
    - **Takes input from sensors (e.g., camera, temperature sensor, user input).**
    - **Updates KB with newly perceived facts.**
4. **Action Selection Module**
    - **Decides next action based on knowledge and inference.**
5. **Learning Mechanism (Optional)**
    - **Improves KB over time by learning from past experiences.**

Working Principle of a Knowledge-Based Agent

A KBA follows a cycle of reasoning and action:

1. **Perceive the environment.**
2. **Update KB with new information.**
3. **Use inference rules to derive conclusions.**
4. **Choose an action based on inferred knowledge.**
5. **Execute the action and update KB accordingly.**



4. Properties of a Knowledge-Based Agent

Property	Description
Explicit Knowledge	Stores facts and rules explicitly.
Logical Reasoning	Uses inference to derive new facts.
Dynamic Updating	Can modify knowledge based on new inputs.
Generalization	Can apply rules to new situations.
Explainability	Can justify its decisions based on known rules.

Real-World Applications of KBAs

- **Chatbots** (e.g., AI assistants using knowledge bases).
- **Medical Diagnosis Systems** (e.g., expert systems like MYCIN).
- **Game AI** (e.g., NPC decision-making using knowledge).
- **Automated Theorem Proving** (e.g., Prolog-based inference systems).
- **Recommendation Systems** (e.g., knowledge-driven movie or book recommendations).

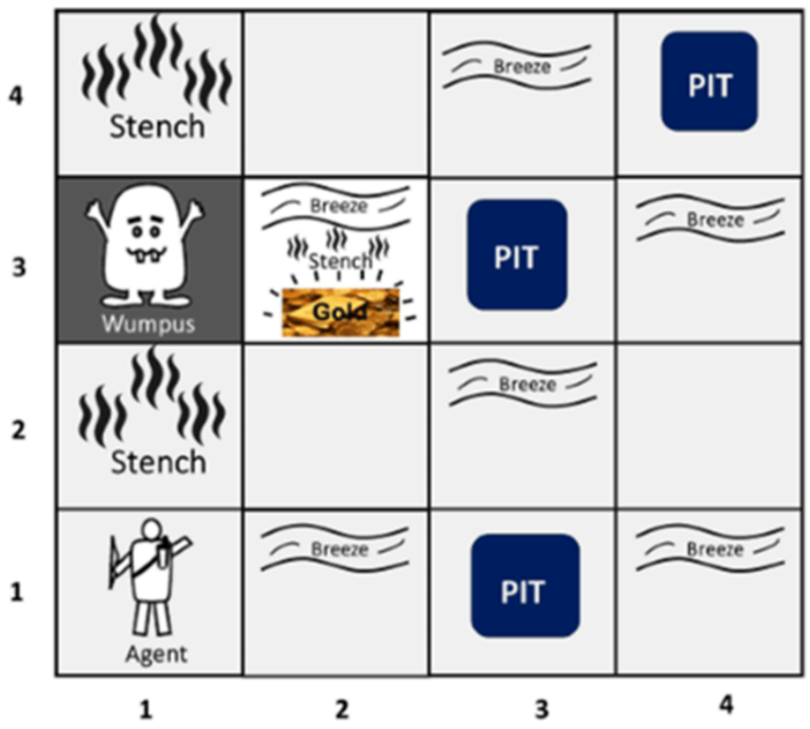
6a Describe syntax and semantics with respect to propositional logic

6b Explain Wumpus World with respect to Artificial Intelligence

Wumpus world:

The Wumpus world is a simple world example to illustrate the worth of a knowledge-based agent and to represent knowledge representation. It was inspired by a video game **Hunt the Wumpus** by Gregory Yob in 1973.

The Wumpus world is a cave which has 4/4 rooms connected with passageways. So there are total 16 rooms which are connected with each other. We have a knowledge-based agent who will go forward in this world. The cave has a room with a beast which is called Wumpus, who eats anyone who enters the room. The Wumpus can be shot by the agent, but the agent has a single arrow. In the Wumpus world, there are some Pits rooms which are bottomless, and if agent falls in Pits, then he will be stuck there forever. The exciting thing with this cave is that in one room

	there is a possibility of finding a heap of gold. So the agent goal is to find the gold and climb out the cave without fallen into Pits or eaten by Wumpus. The agent will get a reward if he comes out with gold, and he will get a penalty if eaten by Wumpus or falls in the pit. Following is a sample diagram for representing the Wumpus world. It is showing some rooms with Pits, one room with Wumpus and one agent at (1, 1) square location of the world.  4 3 2 1 1 2 3 4 Stench Breeze PIT Wumpus Breeze Stench Gold PIT Breeze Stench Breeze Agent Breeze PIT Breeze **There are also some components which can help the agent to navigate the cave. These components are given as follows:** 1. The rooms adjacent to the Wumpus room are smelly, so that it would have some stench. 2. The room adjacent to PITs has a breeze, so if the agent reaches near to PIT, then he will perceive the breeze. 3. There will be glitter in the room if and only if the room has gold. 4. The Wumpus can be killed by the agent if the agent is facing to it, and Wumpus will emit a horrible scream which can be heard anywhere in the cave.		
7a	What is machine learning? Explain with specific examples A computer program is said to learn from experience (E) with respect to some class of tasks (T) and performance measure (P), if its performance at tasks in T, as measured by P, improves with experience E. Machine Learning is about **improving performance (P) on a task (T) based on experience (E).**		

Example 1: Spam Email Detection

- **Task (T): Classifying emails as spam or not spam.**
- **Experience (E): Analyzing a dataset of emails labeled as spam or not spam.**
- **Performance (P): Accuracy in correctly identifying spam emails.**
- **Learning Process:**
  - **The system learns patterns from labeled spam emails.**
  - **It identifies keywords (e.g., "free money", "lottery"), senders, and phrases that indicate spam.**
  - **Over time, as it processes more emails, it improves in identifying spam.**

Example 2: Handwriting Recognition (Digit Classification)

- **Task (T): Recognizing handwritten digits (0-9) from images.**
- **Experience (E): Training on a dataset like MNIST (handwritten digit dataset).**
- **Performance (P): Accuracy in predicting correct digits.**
- **Learning Process:**
  - **The system extracts features (edges, curves, shapes) from images.**
  - **Uses a model (e.g., Neural Networks, SVM) to classify digits.**
  - **With more training data, the accuracy improves.**

A checkers learning problem: I Task T: playing checkers I Performance measure P: percent of games won in the world tournament I Training experience E: games played against itself

7b Explain perspectives and issues with machine learning

Machine Learning (ML) is a field of Artificial Intelligence (AI) that enables computers to **learn from data and improve performance over time without explicit programming**. However, ML comes with multiple **perspectives** and **challenges.**

Perspectives:
The evolution of ML algorithms, such as deep learning and reinforcement learning, has led to state-of-the-art performance in tasks like image recognition, speech processing, and game-playing.

Automating decision-making processes

The ability to process large datasets allows ML systems to uncover patterns and insights that are otherwise impossible to detect manually, enabling data-driven decision-making.

Issues in machine learning

It involves searching a very large space of possible hypotheses to determine one that best fits the observed data and any prior knowledge held by the learner.

Data plays a significant role in the machine learning process. One of the significant issues that machine learning professionals face is the absence of good quality data.

	Many ML models, especially deep learning systems, function as black boxes, making it difficult to interpret how decisions are made. This lack of transparency raises concerns in high-stakes applications like healthcare and law. **ML models can overfit training data, failing to generalize to unseen data. Striking the right balance between model complexity and performance is essential.**		
7c	Explain types of machine learning system Classification based on Whether or not they are trained with human supervision (supervised, unsuper- vised, semi supervised, and Reinforcement Learning) Whether or not they can learn incrementally on the fly (online versus batch learning) Whether they work by simply comparing new data points to known data points, or instead detect patterns in the training data and build a predictive model. Supervised/unsupervised learning Batch or online based learning Model instance based learning		
8a	Describe the main challenges of Machine learning The quality of the learning systems depends on the quality of data. Some of the challenges with this are: ◦Problems: Problems have to be well posed. ◦Insufficient Quantity of Training Data ◦Nonrepresentative Training Data ◦Poor-Quality Data ◦Irrelevant Features ◦Overfitting the Training Data ◦High computational power required ◦Complexity of algorithms ◦Bias/variance		
8b	Explain (i) Find S Algorithm (ii) Candidate Elimination Algorithm The S algorithm, also known as the Find-S algorithm, is a machine learning algorithm that seeks to find a maximally specific hypothesis based on labeled training data. It is a fundamental technique in machine learning that aims to discover a generalized hypothesis from a given set of training data. It is commonly used in concept learning tasks, where the goal is to learn a concept or rule from a set of positive and negative examples. The algorithm starts with the most specific hypothesis and generalizes it by incorporating positive examples. It ignores negative examples during the learning process		

workings of the algorithm:
Initialization − The algorithm starts with the most specific hypothesis, denoted as h. This initial hypothesis is the most restrictive concept and typically assumes no positive examples. It may be represented as h = <∅, ∅, ..., ∅>, where ∅ denotes "don't care" or "unknown" values for each attribute.

Iterative Process − The algorithm iterates through each training example and refines the hypothesis based on whether the example is positive or negative.

For each positive training example (an example labeled as the target class), the algorithm updates the hypothesis by generalizing it to include the attributes of the example. The hypothesis becomes more general as it covers more positive examples.

For each negative training example (an example labeled as a non-target class), the algorithm ignores it as the hypothesis should not cover negative examples. The hypothesis remains unchanged for negative examples.

Generalization − After processing all the training examples, the algorithm produces a final hypothesis that covers all positive examples while excluding negative examples. This final hypothesis represents the generalized concept that the algorithm has learned from the training data.

Candidate elimination Algorithm:

This algorithm computes the version space by the combination of two cases

1.Specific to general learning – Generalize S to include the positive examples

2.General to specific learning – Specialize G to exclude negative examples.

The algorithm defines two boundaries:

1.General boundary: A set of hypotheses that are most general

2.Specific boundary: A set of hypothesis that are the most specific

The algorithm works as follows:

1. Initialization:

- Initialize the specific hypothesis (S) to the first positive example.
- Initialize the general hypothesis (G) to the most general hypothesis (all attributes are “?”).

2. Iterate through examples:

- For each positive example, generalize the specific hypothesis to cover the example.
- For each negative example, specialize the general hypotheses to exclude the example.

3. Refinement:

- Remove hypotheses from G that are more specific than S.

9a Explain working with real data and get the data.

When you are learning about Machine Learning it is best to actually experiment with real-world data, not just artificial datasets. Fortunately, there are thousands of open datasets to choose from, ranging across all sorts of domains.
Popular open data repositories:
—UC Irvine Machine Learning Repository
—Kaggle datasets
—Amazon’s AWS datasets
• Meta portals (they list open data repositories):
—http://dataportals.org/
—http://opendatamonitor.eu/
—http://quandl.com/
• Other pages listing many popular open data repositories:
—Wikipedia’s list of Machine Learning datasets
—Quora.com question

	—Datasets subreddit Real data tends to be noisy, complex, and full of nuances, and therefore introduces a lot of obstacles between you and the question you are trying to answer as a data scientist. Real data provides a more accurate basis for building models and drawing conclusions. Models trained on real data are more likely to generalize well to unseen data because they have been exposed to the same types of variations and uncertainties present in the real world.Whereas the models trained on artificial datasets will stay within their own walls of ignorance and will perform poorly for the outside world. In typical environments your data would be available in a relational database (or some other common datastore) and spread across multiple tables/documents/files. To access it, you would first need to get your credentials and access authorizations,10 and familiarize yourself with the data schema. We can take a look at the data after fetching the data using head() method of Pandas. The info() method is useful to get a quick description of the data, in particular the total number of rows, and each attribute's type and number of non-null values The describe() method shows a summary of the numerical attributes . Another quick way to get a feel of the type of data you are dealing with is to plot a histogram for each numerical attribute. A histogram shows the number of instances (on the vertical axis) that have a given value range (on the horizontal axis). You can either plot this one attribute at a time, or you can call the hist() method on the whole dataset, and it will plot a histogram for each numerical attribute.		
9b	Write a note on Launch, Monitor and Maintain your system. need to write monitoring code to check your system's live performance at regular intervals and trigger alerts when it drops. This is important to catch not only sudden breakage, but also performance degradation. This is quite common because models tend to "rot" as data evolves over time, unless the models are regularly trained on fresh data. Evaluating your system's performance will require sampling the system's predictions and evaluating them. This will generally require a human analysis. These analysts may be field experts, or workers on a crowdsourcing platform. You should also make sure you evaluate the system's input data quality. Sometimes performance will degrade slightly because of a poor quality signal (e.g., a malfunctioning sensor sending random values, or another team's output becoming stale), but it may take a while before your system's performance degrades enough to trigger an alert. If you monitor your system's inputs, you may catch this earlier. Monitoring the inputs is particularly important for online learning systems. Finally, you will generally want to train your models on a regular basis using fresh data. You should automate this process as much as possible. If you don't, you are very likely to refresh your model only every six months (at best), and your system's performance may fluctuate severely over time. If your system is an online learning system, you should make sure you save snapshots of its state at regular intervals so you can easily roll back to a previously working state.		
10 a	Describe the steps involved in preparing the data for the machine learning model. Instead of just preparing data manually, we should write functions to do that, for several good		

	reasons: • This will allow you to reproduce these transformations easily on any dataset (e.g., the next time you get a fresh dataset). • You will gradually build a library of transformation functions that you can reuse in future projects. • You can use these functions in your live system to transform the new data before feeding it to your algorithms. • This will make it possible for you to easily try various transformations and see which combination of transformations works best. Data Cleaning Most Machine Learning algorithms cannot work with missing features.. To handle missing data we have three options: • Get rid of the corresponding districts. • Get rid of the whole attribute. • Set the values to some value (zero, the mean, the median, etc.). We can accomplish these easily using DataFrame's dropna(), drop(), and fillna() methods. Handling Text and Categorical Attributes: Most Machine Learning algorithms prefer to work with numbers anyway, so to convert these categories from text to numbers. For this, we can use Scikit-Learn's Ordina lEncoder class		
10 b	Explain MINST with respect to machine learning. MINST is a set of 70,000 small images of digits handwritten by high school students and employees of the US Census Bureau. Each image is labeled with the digit it represents. This set has been studied so much that it is often called the "Hello World" of Machine Learning: whenever people come up with a new classification algorithm, they are curious to see how it will perform on MNIST. Whenever someone learns Machine Learning, sooner or later they tackle MNIST. There are 70,000 images, and each image has 784 features. This is because each image is 28×28 pixels, and each feature simply represents one pixel's intensity, from 0 (white) to 255 (black). The MNIST dataset is actually already split into a training set (the first 60,000 images) and a test set (the last 10,000 images): The training set is already shuffled for us, which is good as this guarantees that all cross-validation folds will be similar (you don't want one fold to be missing some digits). Moreover, some learning algorithms are sensitive to the order of the training instances, and they perform poorly if they get many similar instances in a row. Shuffling the dataset ensures that this won't happen.		