Solution of Past Paper

Artificial Intelligence

DC-324

Q.1 Write short notes on the following: (6 x 5 = 30 marks)

Question # 01

Define Inference Techniques. (5 marks)

Inference Techniques refer to the methods or strategies used to draw conclusions from data, knowledge bases, or premises in various fields, especially in artificial intelligence (AI) and logic. These techniques are crucial for enabling systems to make decisions, solve problems, or predict outcomes based on the given information.

Key Types of Inference Techniques include:

1. **Deductive Inference:** This technique involves reasoning from general premises to specific conclusions. If the premises are true, the conclusion must be true. It's commonly used in formal logic and mathematics.
2. **Inductive Inference:** Inductive reasoning involves making generalized conclusions based on specific instances or observations. While the conclusions may be probable, they are not guaranteed to be true.
3. **Abductive Inference:** This technique is used to form the best explanation from incomplete or ambiguous data. It's often used in diagnostic applications, where the goal is to find the most likely cause of a given set of symptoms.
4. **Probabilistic Inference:** This method involves calculating the likelihood of various outcomes based on known probabilities. It's widely used in statistical models and machine learning.

Question # 02

What is Heuristics? How to use it in problem-solving? (5 marks)

Heuristics are problem-solving strategies or approaches that use practical methods or shortcuts to produce solutions that may not be perfect but are sufficient for reaching an immediate goal. They are often used when facing complex problems where finding an optimal solution is impractical or time-consuming. Heuristics rely on experience, intuition, and common sense to simplify decision-making processes.

Using Heuristics in Problem-Solving:

1. **Identify the Problem:**
    - Clearly define the problem you are trying to solve. Understanding the problem's nature is the first step in applying heuristics.
2. **Use Rule of Thumb:**
    - Apply a general rule or a common strategy that is known to produce good results in similar situations. For example, in a large search problem, you might prioritize areas that have historically yielded the best results.
3. **Break Down the Problem:**
    - Simplify the problem by breaking it into smaller, more manageable parts. Address each part separately, using heuristic methods such as trial and error or pattern recognition.
4. **Look for Analogies:**
    - Find similarities between the current problem and past problems. Applying solutions that worked in similar situations can be a useful heuristic approach.
5. **Prioritize and Focus:**
    - Focus on the most critical aspects of the problem that will have the greatest impact on the solution. This involves recognizing which parts of the problem can be ignored or approximated.

Question # 03

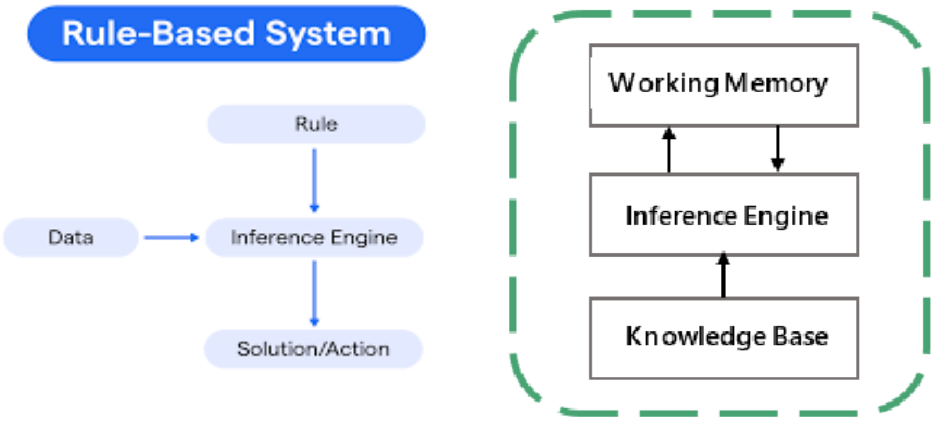
Describe different categories of Expert Systems. Explain only one in detail. (5 marks)

Expert systems are AI programs that mimic the decision-making abilities of a human expert. They are categorized into:

- **Rule-based systems**: Use if-then rules to derive conclusions.
- **Frame-based systems**: Use structured frameworks to represent knowledge.
- **Fuzzy logic systems**: Handle uncertain or imprecise information.
- **Neural network-based systems**: Learn from data to make decisions.

Detailed Explanation (Rule-based systems):

- Rule-based systems rely on a set of if-then rules that represent knowledge in a specific domain. These rules are used to infer conclusions based on the input data. The inference engine applies these rules to the known facts to derive new information or decisions.



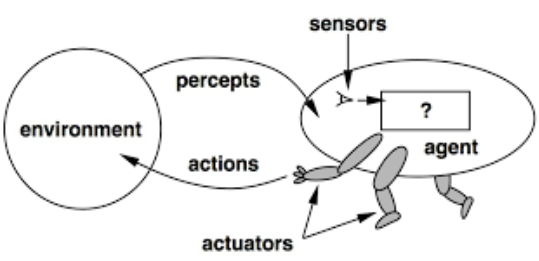
Question # 04

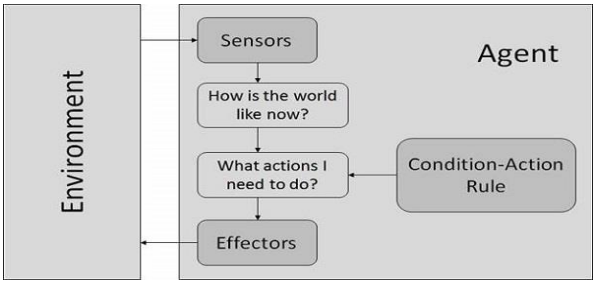
What are Rational Agents? Describe briefly. (5 marks)

A rational agent is an entity that acts to achieve the best possible outcome or, when there is uncertainty, the best expected outcome based on the information it possesses. It perceives its environment through sensors and acts upon that environment using actuators. Rationality is defined in terms of the agent's performance measure, which evaluates the success of its actions in fulfilling its goals.

Key aspects of a rational agent include:

1. **Perception**: Gathers information from the environment.
2. **Performance Measure**: A criterion for success.
3. **Actions**: Acts to maximize the performance measure.
4. **Rationality**: The agent selects actions that maximize its performance based on the available knowledge.





Question # 05

What is a Bidirectional Search? How is it useful? (5 marks)

Bidirectional Search is a graph search algorithm that simultaneously searches from both the initial state (start node) and the goal state (goal node) with the aim of meeting in the middle. This approach reduces the search space, making it more efficient than traditional unidirectional search methods, particularly in large or complex graphs.

How It Works:

- **Forward Search:** Starts from the initial state and explores the graph toward the goal state.
- **Backward Search:** Starts from the goal state and explores the graph toward the initial state.
- **Meeting Point:** The search stops when the two searches meet at a common node, indicating that a path from the start to the goal has been found.

Benefits:

1. **Reduced Search Space:** Instead of searching through all nodes from start to goal, bidirectional search only needs to explore half the nodes from both directions, reducing the total number of nodes explored.
2. **Faster Execution:** Since it reduces the search space, the algorithm often finds the shortest path faster than unidirectional search methods, especially in large graphs.
3. **Memory Efficiency:** By limiting the number of nodes explored, bidirectional search can be more memory efficient, particularly when combined with strategies like BFS (Breadth-First Search) in each direction.

Example Use Case:

Bidirectional search is particularly useful in scenarios such as pathfinding in maps (e.g., finding the shortest route between two cities) or solving puzzles like the 15-puzzle, where the start and goal states are well-defined.

Question # 06

What is Iterative Deepening Depth-First Search? How is it better than Depth-Limited Search? (5 marks)

Iterative Deepening Depth-First Search (IDDFS) is a search algorithm that combines the advantages of both Depth-First Search (DFS) and Breadth-First Search (BFS). It repeatedly applies a Depth-Limited Search (DLS) with increasing depth limits, starting from zero and incrementing by one at each iteration, until the solution is found or a specified depth limit is reached.

How IDDFS works:

1. **Initial Depth-Limited Search:** Start with a depth limit of 0 and perform a DFS.
2. **Increment Depth:** If the solution is not found, increase the depth limit by 1 and perform another DFS, but with the new depth limit.
3. **Repeat:** Continue this process until the solution is found.

Advantages over Depth-Limited Search (DLS):

1. **Completeness:** IDDFS is complete, meaning it is guaranteed to find a solution if one exists. DLS is not complete because it might miss solutions if they are deeper than the depth limit.
2. **Optimality:** IDDFS finds the shortest path to a solution, just like BFS, because it explores all nodes at a given depth before going deeper. DLS, on the other hand, might find a solution that is not optimal if the depth limit is set too high.
3. **Memory Efficiency:** IDDFS uses less memory compared to BFS. Since each iteration of DFS in IDDFS only stores nodes up to the current depth limit, it requires less memory, similar to DFS.
4. **Combines Benefits:** IDDFS combines the depth-first's low memory usage with the breadth-first's completeness and optimality, making it an effective algorithm for many search problems.

Q.2 Answer the following questions: (3 x 10 = 30 marks)

Question # 01

What is a Neural Network? Explain with an example. (10 marks)

A neural network is a computational model inspired by the way biological neural networks in the human brain process information. It is composed of a large number of highly interconnected processing elements known as neurons, which work together to solve specific problems. Neural networks are particularly useful for tasks that involve pattern recognition, classification, and prediction, and they are the foundation of deep learning.

Structure of Neural Networks:

A basic neural network consists of three types of layers:

1. **Input Layer:** This layer receives the input features of the data. Each neuron in this layer represents an input variable, and the number of neurons in the input layer corresponds to the number of features in the dataset.
2. **Hidden Layer(s):** These layers lie between the input and output layers. They perform computations and transformations on the input data by applying weights to inputs, summing them, and passing them through an activation function. A neural network can have one or multiple hidden layers, which allow it to learn complex patterns.
3. **Output Layer:** This layer provides the final output of the network, which can be a classification label, a regression value, or other outcomes depending on the task. The number of neurons in this layer corresponds to the number of possible output classes or the dimensionality of the output.

How It Works:

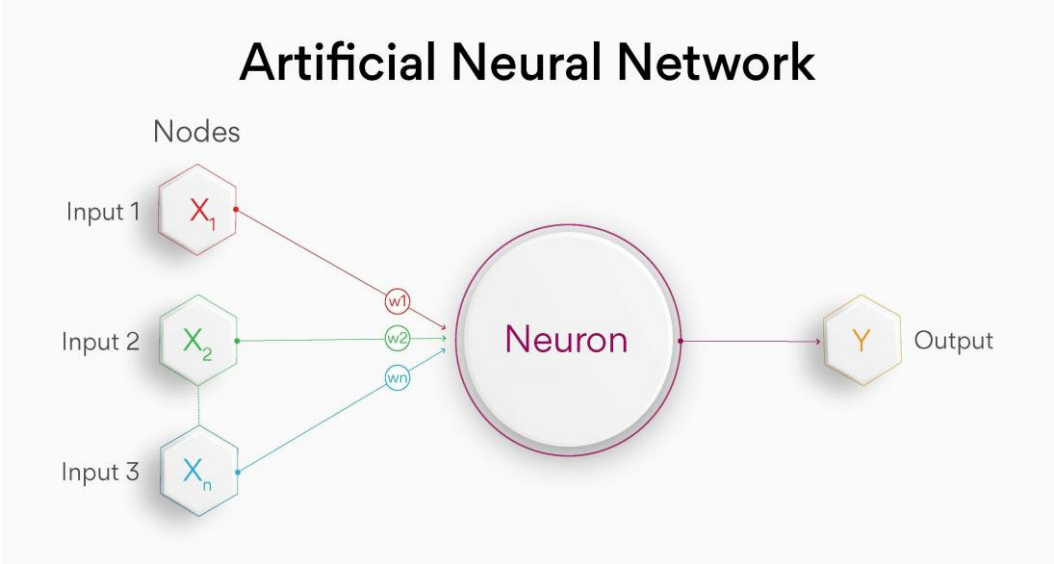
- **Forward Propagation:** During the forward propagation phase, input data is fed into the network, and it passes through each layer of the network. Each neuron in the network calculates a weighted sum of its inputs, applies an activation function, and produces an output. This process continues until the data reaches the output layer.
- **Activation Functions:** These are mathematical functions applied to the output of each neuron to introduce non-linearity into the model, which is crucial for learning complex patterns. Common activation functions include the sigmoid function, hyperbolic tangent (tanh), and rectified linear unit (ReLU).
- **Backpropagation and Training:** The network is trained using a process called backpropagation, which adjusts the weights of the connections to minimize the difference between the predicted output and the actual output (i.e., minimizing the loss function). The training process involves feeding input data into the network, comparing the output to the true labels, calculating the error, and propagating this error backward through the network to update the weights using gradient descent.

Example:

Consider a neural network designed for image classification, such as identifying handwritten digits in the MNIST dataset. The input layer would consist of 784 neurons (one for each pixel in a 28x28 grayscale image). The network might have a few hidden layers, each with several hundred neurons, which learn to recognize patterns like edges, shapes, and textures in the images. Finally, the output layer

would have 10 neurons, each representing a digit (0-9). The network is trained on thousands of labelled images, adjusting its weights to minimize the error in digit recognition. Once trained, the network can accurately classify new, unseen images of digits.

Neural networks have become fundamental tools in many AI applications, including natural language processing, speech recognition, and autonomous vehicles, due to their ability to learn and generalize from large datasets.



Question # 02

What is a Genetic Algorithm? Explain with an example.

(10 marks)

A Genetic Algorithm (GA) is an optimization and search heuristic inspired by the process of natural selection in biological evolution. It is used to find approximate solutions to optimization and search problems. GA's are particularly useful when the search space is large, complex, or poorly understood, and when the objective function is noisy or discontinuous.

Components of a Genetic Algorithm:

1. **Population:** A GA starts with a population of potential solutions, called individuals or chromosomes. Each individual is typically represented by a binary string, though other encodings (e.g., real numbers) are also used. The population represents different points in the search space.

2. **Fitness Function:** The fitness function evaluates how good each individual is at solving the problem. It assigns a fitness score based on how close the individual is to the optimal solution. The goal is to maximize (or minimize) this fitness score.
3. **Selection:** Individuals are selected from the current population to be parents for the next generation. The selection process favour individuals with higher fitness scores, increasing the likelihood that their genetic material (encoded solutions) will be passed on to the next generation.
4. **Crossover (Recombination):** During crossover, pairs of parents are selected, and their genetic material is combined to produce offspring. The crossover operator might, for example, swap sections of the parents' binary strings to create new individuals. This process introduces new combinations of traits into the population.
5. **Mutation:** Mutation introduces random changes to individual genes in the offspring, ensuring genetic diversity within the population. For example, in a binary-encoded individual, a mutation might flip a bit from 0 to 1. Mutation helps the GA to explore the search space more thoroughly and avoid local optima.
6. **Replacement:** The new generation of offspring replaces the old population. The GA repeats the selection, crossover, mutation, and replacement steps for many generations until it finds an optimal or satisfactory solution.

Example:

Suppose we want to use a GA to optimize the design of an antenna for a spacecraft. The design problem is highly complex due to the numerous parameters involved, such as the length, width, and material of the antenna, and how these parameters interact with each other to affect performance (e.g., signal strength, bandwidth).

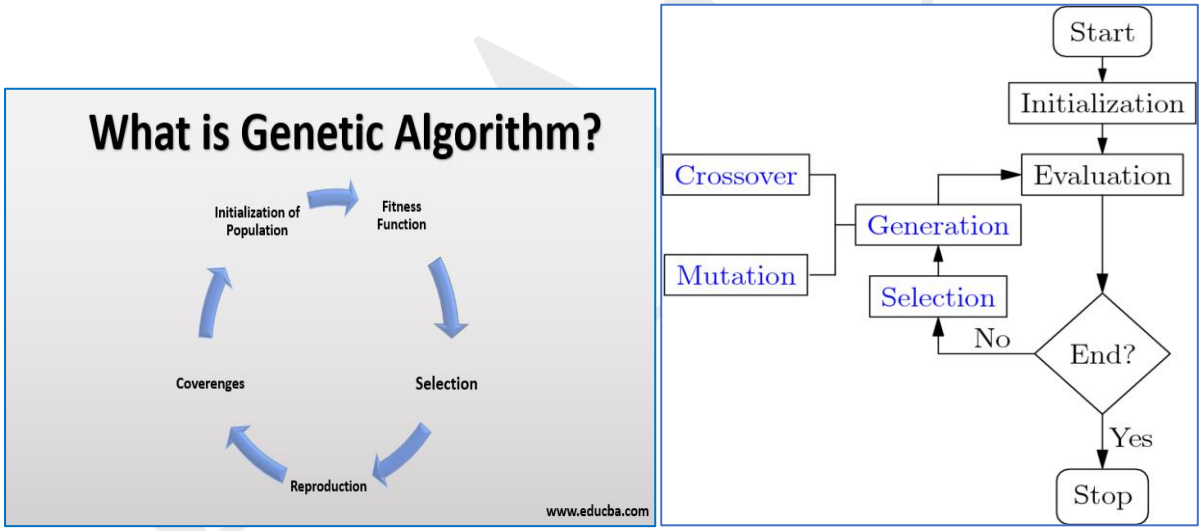
1. **Encoding:** Each individual in the population represents a potential antenna design. The design parameters are encoded as a binary string. For example, if there are 10 parameters, each represented by 5 bits, then each individual might be a 50-bit binary string.
2. **Fitness Function:** The fitness function evaluates how well each antenna design meets the objectives (e.g., maximizing signal strength and minimizing interference). The fitness score might be a weighted sum of these objectives, penalizing designs that do not meet certain constraints.
3. **Selection:** The algorithm selects the top-performing designs based on their fitness scores to serve as parents for the next generation.

4. **Crossover and Mutation:** Crossover combines the traits of two parent designs to create a new design, while mutation randomly alters some design parameters, introducing new potential solutions into the population.
5. **Evolution:** Over many generations, the GA evolves the population, gradually improving the antenna designs. The best designs from each generation are carried forward, and after sufficient iterations, the GA produces a highly optimized antenna design.

Applications of Genetic Algorithms: GA's are used in a wide range of fields, including engineering design, robotics, financial modelling, bioinformatics, and artificial intelligence. They are particularly effective for solving problems where traditional optimization techniques struggle, such as in highly non-linear, multi-modal, or poorly understood environments.

Diagram:

Here is the diagram of Generic Algorithm Process:



Question # 03

What is the Task Environment of an Agent? Describe different task environments briefly. (10 marks)

In artificial intelligence (AI), the task environment refers to the external setting in which an agent operates and interacts. It encompasses everything the agent must consider to make decisions and achieve its goals. Understanding the task environment is crucial for designing intelligent agents because it dictates the

agent's perceptual and action capabilities, as well as the challenges it must overcome.

Key Characteristics of Task Environments:

1. **Fully Observable vs. Partially Observable:**
    - **Fully Observable:** In a fully observable environment, the agent has access to the complete state of the environment at all times. This allows the agent to make informed decisions without any uncertainty about the current situation. For example, in a game like chess, the entire board is visible, so the environment is fully observable.
    - **Partially Observable:** In a partially observable environment, the agent only has partial information about the state. This could be due to sensors' limitations or inherent uncertainties in the environment. For example, a self-driving car may not have complete visibility of the road due to fog or other vehicles obstructing its view.
2. **Deterministic vs. Stochastic:**
    - **Deterministic:** In a deterministic environment, the next state is entirely determined by the current state and the agent's action. There is no randomness involved. For instance, a mathematical puzzle where the outcome is predictable and depends solely on the agent's moves is deterministic.
    - **Stochastic:** In a stochastic environment, there is randomness, meaning the same action in the same state can lead to different outcomes. For example, in stock market trading, the outcome of an investment decision is influenced by random market fluctuations, making the environment stochastic.
3. **Static vs. Dynamic:**
    - **Static:** In a static environment, the world does not change while the agent is deliberating. The agent can take its time to decide without worrying about changes in the environment. An example of a static environment is a crossword puzzle, where the puzzle does not change as the solver thinks.
    - **Dynamic:** In a dynamic environment, the environment changes over time, and these changes can happen independently of the agent's actions. For instance, in real-time strategy games, other players or the environment may change during the agent's decision-making process.
4. **Discrete vs. Continuous:**
    - **Discrete:** A discrete environment has a finite number of states and actions. The agent's actions lead to transitions between distinct

states. For example, in a board game like tic-tac-toe, the board has a finite number of configurations, and the player's actions are discrete moves.
    - **Continuous:** A continuous environment has an infinite number of possible states and actions. For example, a robot navigating through physical space may have infinitely many possible positions and actions, making the environment continuous.
5. **Episodic vs. Sequential:**
    - **Episodic:** In an episodic environment, the agent's experiences are divided into episodes, and each episode is independent of the others. The agent's action in one episode does not affect the next. For instance, in image recognition tasks, the classification of one image does not impact the classification of another.
    - **Sequential:** In a sequential environment, the current decision may affect future decisions. The agent's actions have long-term consequences. For example, in a game of chess, each move impacts the entire game, so the environment is sequential.

Examples of Task Environments:

1. **Autonomous Vehicles:** An autonomous vehicle operates in a highly dynamic, partially observable, and continuous environment. The car must navigate through traffic, avoid obstacles, and respond to changes in real-time, all while dealing with incomplete information (e.g., hidden pedestrians).
2. **Robotic Vacuum Cleaners:** A robotic vacuum operates in a dynamic, partially observable, and discrete environment. It must navigate a home environment where objects and people can move, requiring the robot to adjust its path while having limited information about the entire layout.
3. **Medical Diagnosis Systems:** These systems work in a partially observable and stochastic environment. The system must make diagnoses based on incomplete patient information and deal with the uncertainty of symptoms that could point to multiple diseases.

Designing Agents for Task Environments:

When designing an agent, understanding the task environment allows AI developers to choose the right strategies and algorithms. For instance, in a fully observable and deterministic environment, a rule-based system might suffice. However, in a partially observable, stochastic, and dynamic environment, the agent might require more advanced techniques such as machine learning, probabilistic reasoning, and real-time decision-making.

Understanding these characteristics helps in crafting intelligent agents that can function effectively in their intended environments, ensuring that they are both robust and adaptable to the challenges they will face.