EXP 1

The **PEAS** representation framework helps to understand the design and evaluation of AI systems by breaking down the components involved.

**Performance Measure:** How the AI system's success is measured, often by an external observer or evaluator. For ex., in a medical diagnosis AI system, it could be the accuracy of diagnoses compared to human experts.

**Environment:** This encompasses the surroundings and context in which the AI system operates. In the case of medical diagnosis, the environment includes patient data, medical records, diagnostic tests, and medical knowledge databases.

**Actuators:** These are the actions that the AI system can perform within its environment. For a medical diagnosis AI, actuators could include recommending treatment plans, suggesting further tests, or providing diagnostic explanations to medical professionals.

**Sensors:** Sensors enable the AI system to observe and gather information from its environment. In medical diagnosis, sensors could include patient vital signs, lab test results, medical imaging scans, and textual medical records.

EXP 2

**Breadth First Search** (Queue) (BFS) algorithm is a method for exploring trees of nodes.

1. **Enqueue root node:** Begin by placing initial or root node into a queue DS.
2. **Dequeue a node and examine it:** Remove node from front of the queue. If desired element is found in this node, search terminates, & a result is returned. Otherwise, enqueue any successors of current node that haven’t been discovered.
3. **Check if the queue is empty:** If it is, it means every node in the graph has been examined without finding the desired element, the search terminates.
4. **Repeat:** If the queue is not empty, return to step 2 and continue the process.

BFS explores the search space in a level-by-level manner, ensuring that all nodes at a given depth are examined before moving deeper into the search tree. This approach guarantees that BFS will find the shortest path (priority) to a goal node if one exists.

EXP 3

**A\* heuristic search** algorithm that evaluates nodes by combining the actual cost to reach (g(n)) and the estimated cost (h(n)). **f(n) = g(n) + h(n)**

f(n): estimated cost of the cheapest solution through node n.

A\* is effective as it balances cost of path from start (g(n)) & estimated cost to goal (h(n)).

Requirements to work optimally: choose Admissible heuristic (h(n)) that never overestimates the cost to reach the goal. An admissible heuristic ensures completeness & optimality.

A\* minimizes the total estimated solution cost. This approach combines the advantages of both strategies, resulting in a highly efficient search algorithm.

In summary, A\* algorithm utilizes a heuristic function that estimates the **shortest distance from the initial state to the closest goal state**, incorporating both the traveled distance and the predicted distance ahead to guide the search effectively.

EXP 4

The methodology for game-playing agents:

1. **Initial state:** Identifies the board position and the player to move.

2. **Successor function:** Returns legal moves and resulting states.

3. **Terminal test:** Determines when the game ends.

4. **Utility function:** Assigns numeric values to terminal states (+1=win, -1=loss, 0=draw).

The game tree is constructed based on the initial state and legal moves, with each level representing a ply (half a game turn). The Min-Max algorithm operates on this tree:

**Min-Max Algorithm:** Evaluates each node in the game tree recursively. Nodes are labeled as Max nodes if the player aims to maximize outcome (Agent) (best possible move) or Min nodes if the player aims to minimize outcome (Opponent) (worst possible move). Each node is assigned Min-Max value, representing utility of node assuming both players play optimally.

The Min-Max value is calculated recursively using the formula:

Min-Max Value = Utility value if node is terminal

Max of Min-Max value of successor nodes if Max node

Min of Min-Max value of successor nodes if Min node

Implementation details involve storing the game tree in memory, designing a utility function tailored to the game, and assigning utility values to leaf nodes based on game outcomes. For zero-sum games, utility values are +1 for agent wins, 0 for draws, and -1 for opponent wins. In non-zero sum games, utility values are assigned based on the agent's perspective. Utility values propagate up the tree, choosing Max for Agent-ply and Min for Opponent-ply until the root node is reached.

EXP 5

**Prolog** is a logic programming language associated with AI and computational linguistics. It is based on a formal system called first-order logic, or predicate logic, which allows for the representation of facts, rules, and queries in a declarative manner.

Prolog is commonly used in various fields such as AI, natural language processing, expert systems, and automated reasoning. It is particularly well-suited for tasks involving symbolic computation, rule-based reasoning, and symbolic pattern matching.

1. The PROLOG suite is based on Interpreter. Prolog is typically implemented using an interpreter, which evaluates the logic-based rules and queries in the Prolog program.
2. Facts and rules define predicates. A fact is a statement that is always true. Therefore, there must be at least one fact about each predicate in the Prolog program.
3. In Prolog, variable declaration is not required. Variables are implicitly declared when they are used in predicates or rules.
4. **Fact:** A fact is a statement that is always true. For example: father(john, mary).
5. **Predicate:** A predicate is a rule or statement that can be true or false depending on the values of its arguments. For example: father(X, Y) :- male(X), parent(X, Y).

EXP 6

The **8-puzzle**: goal is to rearrange a set of numbered tiles on a grid to reach a desired configuration. The puzzle consists of an n x n grid with n^2 - 1 tiles numbered from 1 to n^2 - 1, and one empty space. The objective is to move the tiles into a specified goal state by sliding them into the empty space.

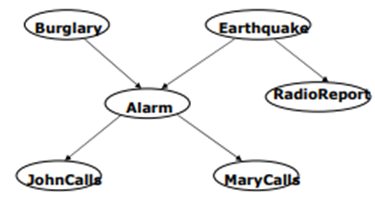
**Rules:**

1. You can only move tiles horizontally or vertically into the empty space.
2. Each move has a cost of 1.
3. The puzzle is solved when the tiles are arranged in the specified goal configuration.

We use A\* search algorithm. The solvepuzzle predicate takes three parameters: the initial state, the goal state, and a variable to return the total cost of reaching the goal state.

The A\* search algorithm expands nodes in the search space while minimizing the total estimated cost of reaching the goal state. It uses a heuristic function, such as the Manhattan distance, to guide the search towards the goal.

EXP 7

**Bayesian networks** are graphical models used to represent probabilistic relationships between variables.

Nodes represent random variables.

Edges represent probabilistic dependencies b/w variables.

Conditional probability distributions are associated with each node, capturing probability of node given its parents' values.

Bayesian Network Diagram:

"Burglary" and "Earthquake" are independent.

"Burglary" and "Radio Announcement" are independent given "Earthquake".

**Netica** offers various **features** such as:

1. Graphical interface for constructing Bayesian networks.
2. Ability to define nodes and their dependencies.
3. Functions for specifying conditional probability distributions.
4. Tools for performing inference and probabilistic reasoning.
5. Options for sensitivity analysis and model validation.
6. Support for visualization of network structures and probability distributions.
7. JohnCalls is independent of Burglary, given Alarm. False: It depends on Burglary indirectly through Alarm. If Alarm is raised due to Burglary, JohnCalls may be likely.
8. Burglary is independent of Earthquake (not knowing Alarm), but Burglary and Earthquake become dependent given Alarm. True: If we don't know the state of Alarm, Burglary and Earthquake are independent. However, knowing Alarm, they become dependent as Alarm can be triggered by either Burglary or Earthquake.
9. MaryCalls is independent of JohnCalls, given Alarm. True: MaryCalls and JohnCalls may be independent given Alarm. The occurrence of one does not necessarily influence the occurrence of the other, given the presence of Alarm.

EXP 8

**Decision Tree** is a supervised learning algorithm used for classification and regression tasks. It creates a tree-like model of decisions by partitioning the data into subsets based on attribute values. Each internal node represents a feature, each branch represents a decision rule, and each leaf node represents the outcome or target variable.

**Classification:** target variable=categorical, & decision tree predicts class label of input data.

**Regression:** target variable=continuous, & decision tree predicts the value of the input data.

Features of Decision Trees:

1. **Root Node:** Represents the entire dataset.
2. **Splitting:** Process of dividing a node into sub-nodes based on attribute values.
3. **Decision Node:** Sub-node resulting from splitting.
4. **Leaf/ Terminal Node:** Node that does not split further, representing the outcome.
5. **Pruning:** Process of removing sub-nodes to avoid overfitting.
6. **Branch/Sub-Tree:** Subsection of the entire tree.
7. **Parent and Child Node:** Relationship between nodes.

**ID3 Algorithm (Iterative Dichotomiser 3)** is a decision tree algorithm that builds trees using a top-down greedy search approach. It selects attributes based on Information Gain, a measure of entropy reduction. Steps in ID3 Algorithm:

1. Start with the root node.
2. Calculate Information Gain for each attribute.
3. Select the attribute with the highest Information Gain as the decision node.
4. Split the dataset based on the selected attribute.
5. Recur on each subset until all attributes are used or entropy is zero.

**Attribute Selection Measures:**

1. **Information Gain:** Measures changes in entropy after dataset segmentation.
2. **Entropy:** Measures impurity in an attribute.
3. **Gini Index:** Measures impurity used in CART algorithm.

**Confusion Matrix:** used to evaluate the classification performance of a model. It provides counts of test records correctly and incorrectly predicted by the model. Accuracy is computed from the confusion matrix to measure the model's effectiveness.

1. Develop a Decision Tree model using the ID3 algorithm.
2. Select attributes based on Information Gain.
3. Split nodes recursively until all attributes are used or entropy is zero.
4. Prune the tree to avoid overfitting.
5. Evaluate the model's performance using a confusion matrix.