**SRM Institute of Science and Technology**

Mode of Exam

**OFFLINE**

**SET D**

**College of Engineering and Technology**

**School of Computing**

**DEPARTMENT OF COMPUTATIONAL INTELLIGENCE**

SRM Nagar, Kattankulathur – 603203, Chengalpattu District, Tamil Nadu

**Academic Year: 2023 - 24 (EVEN)**

**Test: CLAT- 2** **Date: 02.04.2023**

**Course Code & Title: 18CSC305J / Artificial Intelligence** **Duration:** **1 hr 40 mts**

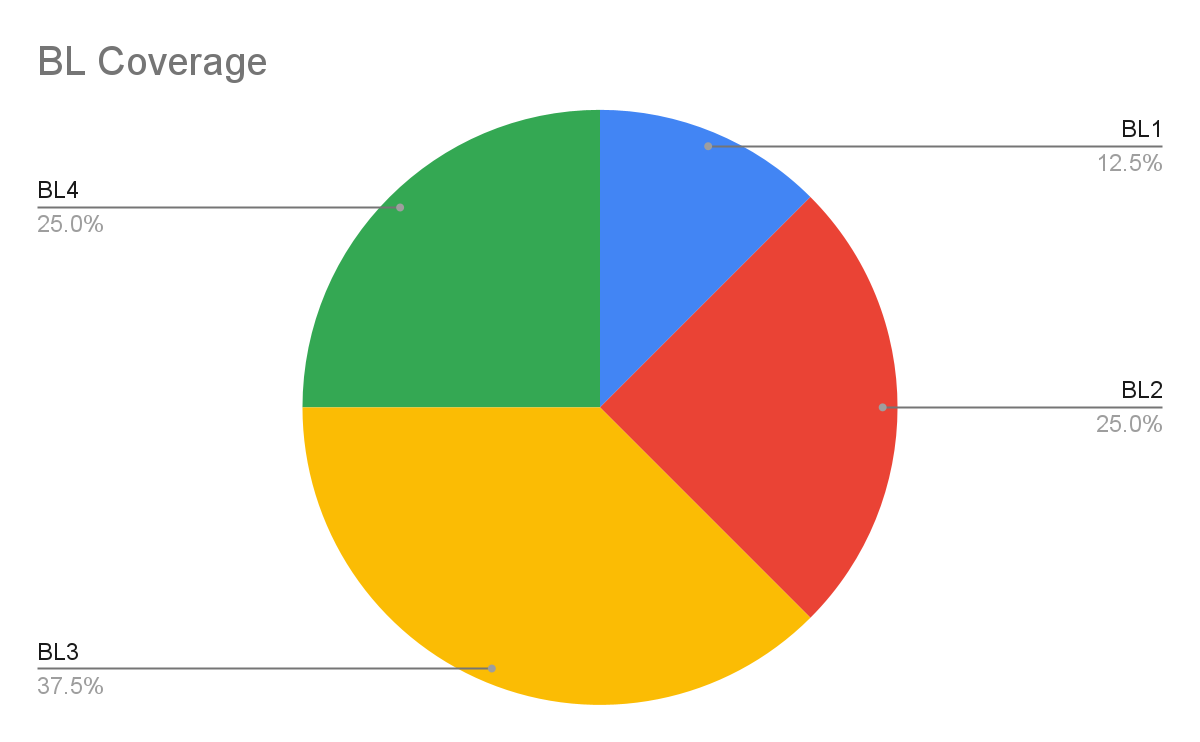
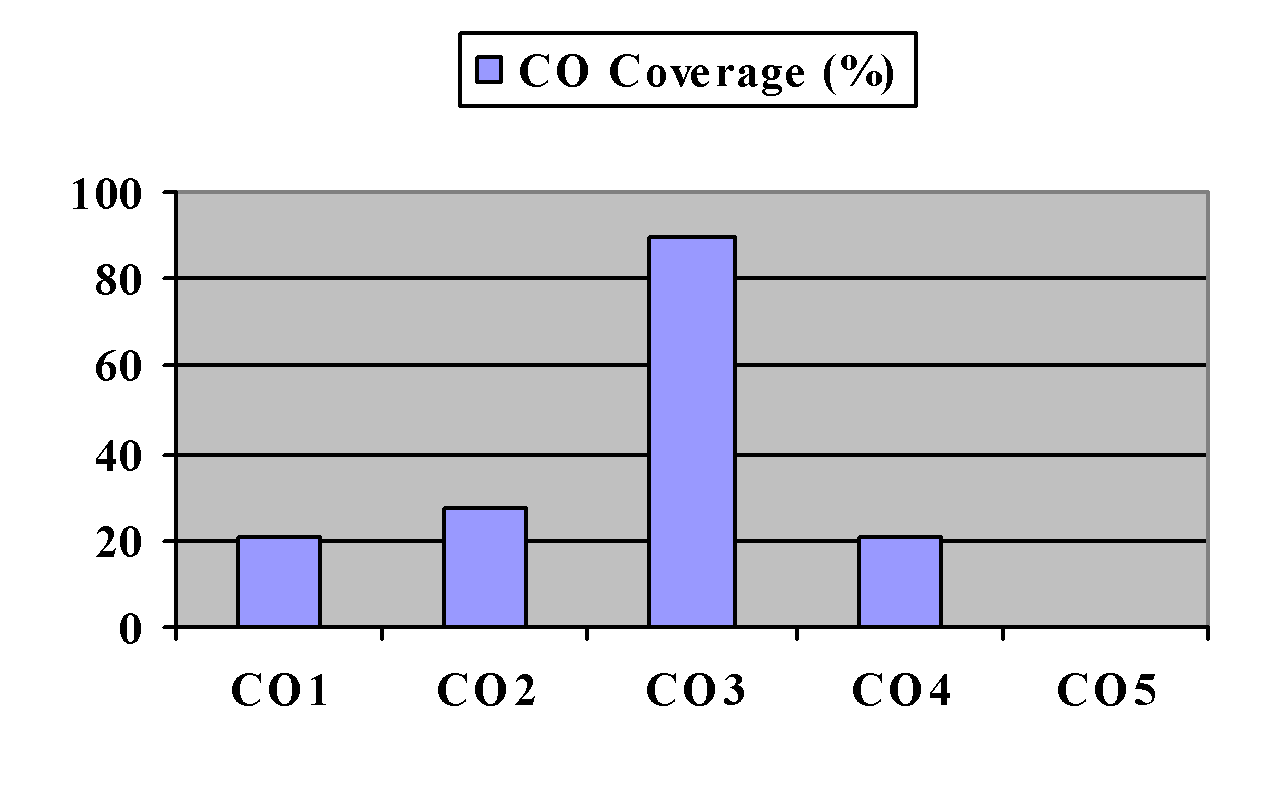
**Year & Sem:** **III / VI** **Max. Marks:** **50**

**Course Articulation Matrix: *(to be placed)***

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| S.No. | Course Outcome | PO1 | PO2 | PO3 | PO4 | PO5 | PO6 | PO7 | PO8 | PO9 | PO10 | PO11 | PO12 |
| 1 | CO1 | 3 | 2 | 3 | - | - | - | - | - | - | - | - | - |
| 2 | CO2 | 3 | 2 | 3 | - | - | - | - | - | - | - | - | - |
| 3 | CO3 | 2 | 3 | 3 | - | - | - | - | - | - | - | - | - |
| 4 | CO4 | 2 | 3 | 2 | - | - | - | - | - | - | - | - | - |
| 5 | CO5 | 2 | 3 | 3 | 2 | - | - | - | - | - | - | - | - |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Part - A**  (5 x 10 = 50 Marks)  Instructions: Answer any 5 Questions | | | | | | |
| Q. No | Question | Marks | BL | CO | PO | PI Code |
|  | Consider the below graph    Apply the following methods to the above graph:   1. Implement uniform cost search, where the start node is “Home” and the goal is “E-square.”   **Ans:**  **Uniform cost search prioritizes expanding nodes with the lowest path cost. In this case, the cost between nodes is represented by the numerical labels on the edges.**  **Initialize an empty frontier (priority queue) and an empty explored set.**  **Add the starting node ("Home") to the frontier with a cost of 0.**  **While the frontier is not empty:**  **Remove the node with the lowest cost from the frontier and add it to the explored set.**  **If the removed node is the goal node ("E-square"), then the search is successful and you can terminate the loop.**  **For all unexpanded neighbor nodes of the removed node:**  **Calculate the tentative cost of reaching the neighbor node by adding the edge cost to the cost of reaching the current node.**  **If the neighbor node is not in the explored set or the tentative cost is lower than the neighbor's current cost in the frontier, then add the neighbor node to the frontier with the calculated tentative cost.**  **Following this algorithm, you'll find the path to "E-square" with a total cost of 24:**  **Home -> Pune Central College (cost 10) -> Agricultural College (cost 3) -> University (cost 11) -> E-square (cost 0**   1. In the absence of the distance parameter, implement DFS and show step-wise how the goal state is reached.   **Ans:**  **DFS explores as deep as possible along a single path before backtracking and exploring another path.**  **Initialize an empty stack and an empty explored set.**  **Push the starting node ("Home") onto the stack.**  **While the stack is not empty:**  **Pop a node from the stack and add it to the explored set.**  **If the popped node is the goal node ("E-square"), then the search is successful and you can terminate the loop.**  **For all unexpanded neighbor nodes of the popped node:**  **If the neighbor node is not in the explored set, push it onto the stack.**  **In this graph, DFS will explore the following path to reach "E-square":**  **Home -> Deep Bunglow Chowk (cost 12) -> Fergusson College (cost 15) -> Symbiosis School (cost 19) -> E-square (cost 0)**   1. In the absence of the distance parameter, implement BFS and show step-wise how the goal state is reached.   **Ans:**  **BFS expands all neighbor nodes of a level before moving to the next level.**  **Initialize an empty queue and an empty explored set.**  **Add the starting node ("Home") to the queue.**  **While the queue is not empty:**  **Remove a node from the queue and add it to the explored set.**  **If the removed node is the goal node ("E-square"), then the search is successful and you can terminate the loop.**  **For all unexpanded neighbor nodes of the removed node:**  **If the neighbor node is not in the explored set or not already in the queue, add it to the queue.**  **Following BFS, you'll find the path to "E-square" with a cost of 24, which is the shortest path:**  **Home -> Pune Central College (cost 10) -> Agricultural College (cost 3) -> University (cost 11) -> E-square (cost 0)**   1. In the absence of the distance parameter, implement DLS and show step-wise how the goal state is reached (where the limit is 3)   **Ans:**  **DLS restricts the depth of exploration to a predefined limit. Here, the limit is set to 3.**  **Perform a depth-first search but keep track of the current depth.**  **If the current depth exceeds the limit, terminate the exploration of that path and backtrack.**  **With a limit of 3, DLS will explore up to depth 3 from the starting node but won't reach the goal "E-square" which resides at a deeper level.**   1. In the absence of the distance parameter, implement IDS and show step-wise how the goal state is reached (where the limit is 3)   **Ans :**  **IDS performs multiple DFS searches with increasing depth limits until the goal is found.**  **For each depth limit (starting from 1):**  **Perform a DFS search with the current depth limit.**  **If the DFS search reaches the goal node, the search is successful and you can terminate the loop.**  **Following IDS with a limit of 3, the first DFS search (with limit 1) won't reach the goal. The second DFS search (with limit 2) will reach "University" but not "E-square".** | 10 | 3 | 2 | 3 | 3.2.1 |
| 2. | The two-player search tree to understand the working of Alpha-beta pruning.    ANS: This tree represents a game where two players take turns making decisions, and each player aims to maximize or minimize their score.  Terminal Nodes (D, E, F, G):  The values for these nodes are already given:  D = 2  E = 3  F = 9  G = 0  Node B (Max Node):  Node B is a Max node, meaning our player wants to maximize their score.  We compare the values of its children (D and E):  max(2, 3) = 3  So, the value of Node B is 3.  Node C (Max Node):  Node C is also a Max node.  We compare the values of its children (F and G):  max(9, 0) = 9  The value of Node C is 9.  Node A (Min Node):  Node A is a Min node, representing the opponent’s move.  We compare the values of its children (B and C):  min(3, 9) = 3  The optimal value for the player who starts (assuming it’s a maximizing player) is 3. | 10 | 3 | 2 | 3 | 3.2.1 |
| 3. | Identify the shortest path using A\* algorithm, and also mention its algorithmic properties. Actions: Horizontal, Vertical, Diagonal  Starting Point: Dog  Goal Point: Bone    ANS:  **Answer:**  This algorithm is commonly used for pathfinding and graph traversal, especially in scenarios where we need to find the shortest path efficiently.  Problem Description:  We have a square grid with various obstacles.  Given a starting cell (marked as “Dog”) and a target cell (marked as “Bone”), we want to reach the target cell as quickly as possible.  The A\* algorithm will guide us toward the optimal path.  A Algorithm Overview:\*  At each step, the A\* algorithm selects the node with the lowest value of “f”, which is the sum of two parameters:  “g”: The movement cost from the starting point to the current cell.  “h”: The estimated movement cost from the current cell to the goal (heuristic).  The algorithm combines these values to prioritize exploration.  Step-by-Step Solution:  Let’s calculate the values for each cell:  Dog (Starting Point):  G = 2 (movement cost from start)  H = 6 (estimated cost to reach the goal)  Adjacent Cells:  “Stick”:  G = 4  H = 6  F = 10 (G + H)  “Stend”:  G = 8  H = 5  F = 13 (G + H)  “Pebbles” (via diagonal move):  G = 6  H = 8  F = 14 (G + H)  Move to “Stick” (lowest F score):  Adjacent cells: “Pole” and “Stend”  Move to “Stend”:  Adjacent cells: “Pole,” “Rope,” and “Pebbles”  Move to “Pole”:  Only adjacent cell: “Rope”  Move to “Rope”:  Only adjacent cell: “Bone” (Goal Point)  Reach the Goal at “Bone”!  Algorithmic Properties of A Algorithm:\*  Admissible:  If a solution exists for the given problem, the first solution found by A\* is an optimal solution.  Complete:  A\* is a complete algorithm, meaning if a solution exists, it will be found in a finite amount of time.  Optimal:  A\* is optimally efficient for a given heuristic. | 10 | 3 | 2 | 3 | 3.2.1 |
| 4. | Explain the main components and workings of a Genetic Algorithm (GA). Provide a detailed solution for a maximization problem using a simple example.  **Solution:**  Genetic Algorithm (GA) is a search heuristic inspired by the process of natural selection and genetics. It's commonly used to generate high-quality solutions to optimization and search problems. Here are the main components and workings of a GA:  Initialization:  A population of individuals (possible solutions) is randomly generated. Each individual typically represents a potential solution to the problem.  Fitness Evaluation:  Each individual in the population is evaluated for its fitness, which represents how good the solution is. This is typically done by a fitness function that quantifies the quality of the solution.  Selection:  Individuals are selected from the current population to form a mating pool, typically based on their fitness. Better individuals have a higher chance of being selected.  Reproduction (Crossover):  Pairs of individuals (parents) from the mating pool are selected to produce offspring through crossover (recombination). This involves exchanging genetic information (typically represented as strings) to create new individuals.  Mutation:  After crossover, some genetic material in the offspring may be randomly altered to introduce diversity and prevent premature convergence to suboptimal solutions.  Replacement:  The new offspring replaces some individuals in the current population, typically through a process such as elitism where the best individuals are preserved.  Termination:  The process continues iteratively until a termination condition is met, such as a maximum number of generations reached or a satisfactory solution found.  Example:  Let's consider a simple maximization problem of finding the maximum value of the function f(x) = x^2 in the range [0, 31]. We will use a binary representation with 5 bits to represent each individual.  Initialization:  Generate an initial population of random binary strings:  Population = [10101, 11011, 00100, 11100, 01010, ...]  Fitness Evaluation:  Evaluate the fitness of each individual using the function f(x) = x^2.  Selection:  Select individuals from the population based on their fitness. Use methods like Roulette Wheel Selection or Tournament Selection.  Reproduction (Crossover):  Perform crossover to produce offspring. For example, select two parents: 10101 and 11011. Perform crossover at a random point to produce two offspring: 10111 and 11001.  Mutation:  Apply the mutation to the offspring. Randomly flip some bits in the offspring strings.  Replacement:  Replace some individuals in the population with the new offspring. Use elitism to ensure that the best individuals are preserved.  Termination:  Repeat the process for a certain number of generations or until a satisfactory solution is found.  By iterating through these steps, the GA evolves a population towards better solutions over successive generations until it converges to an optimal or near-optimal solution for the problem at hand. | 10 | 3 | 2 | 3 | 3.3.1 |
| 5. | A knowledge base has the following statements:   * If there is gas in the tank and the fuel line is okay, then there is gas in the engine; * If there is gas in the engine and a good spark, the engine runs; * If there is power to the plugs and the plugs are clean, a good spark is produced; * If the battery is charged and the cables are okay, then there is power to the plugs.   a. Convert the rules above to CNF using proposition symbols such as GasInTank, FuelLineOK, GasInEngine, etc.  b. Suppose that you are given the fact that there is gas in the tank, the battery is charged, the fuel line and cables are both okay, and the plugs are clean. Using resolution, prove that the engine runs.  **Answer:**  **a. Converting the rules to Conjunctive Normal Form (CNF) using proposition symbols:**  **1. (GasInTank∧FuelLineOK)→GasInEngine**  **(¬GasInTank∨¬FuelLineOK∨GasInEngine)**  **2. (GasInEngine∧GoodSpark)→EngineRuns**  **(¬GasInEngine∨¬GoodSpark∨EngineRuns)**  **3. (PowerToPlugs∧PlugsClean)→GoodSpark**  **(¬PowerToPlugs∨¬PlugsClean∨GoodSpark)**  **4. (BatteryCharged∧CablesOK)→PowerToPlugs**  **(¬BatteryCharged∨¬CablesOK∨PowerToPlugs)**  **b. Proving that the engine runs using resolution:**  **Given:**  **• GasInTank**  **• BatteryCharged**  **• FuelLineOK**  **• CablesOK**  **• PlugsClean**  **To prove:**  **• EngineRuns**  **Using resolution, we start with the negation of what we want to prove and then derive a contradiction:**  **1. ¬EngineRuns (Negation of what we want to prove)**  **2. GasInEngine∨¬GoodSpark∨EngineRuns (From rule 2)**  **3. ¬PowerToPlugs∨¬PlugsClean∨GoodSpark (From rule 3)**  **4. ¬BatteryCharged∨¬CablesOK∨PowerToPlugs (From rule 4)**  **5. GasInTank∨¬FuelLineOK∨GasInEngine (From rule 1)**  **6. GasInEngine (From 5 and given InTank)**  **7. GoodSpark (From 3 and given PlugsClean and PowerToPlugs from 4)**  **8. EngineRuns (From 2, 6, and 7)**  **Since we derived EngineRuns (step 8) without a contradiction, we have proved that the engine runs.** | 10 | 3 | 3 | 3 | 3.3.1 |
|  | An autonomous car is driving on a highway. There are three possible lanes the car can be in: Lane 1 (left), Lane 2 (center), and Lane 3 (right). A camera sensor detects an object in front of the car. Based on the image resolution, the camera sensor believes there's a 60% chance the object is on Lane 2 and a 20% chance it's on each of Lanes 1 and 3 (Left and Right).  A LiDAR sensor also detects the object. Due to limitations in LiDAR's ability to differentiate lanes at long distances, it believes there's a 40% chance the object is in Lane 2 and a 30% chance it's in each of the other two lanes (Lane 1 and Lane 3).  Use the Dempster-Shafer Theory to determine the car's belief in the object's location (Lane 1, Lane 2, or Lane 3).  Ans:  1. Frame of Discernment (Theta):  Theta = {Lane 1, Lane 2, Lane 3}  This represents all possible locations for the object.  2. Belief Functions:  Camera Sensor (Bel\_Camera):  Bel\_Camera(Lane 1) = 0.2 (20% chance)  Bel\_Camera(Lane 2) = 0.6 (60% chance)  Bel\_Camera(Lane 3) = 0.2 (20% chance)  Bel\_Camera ({}) = 0 (Empty set, no opinion)  LiDAR Sensor (Bel\_LiDAR):  Bel\_LiDAR(Lane 1) = 0.3 (30% chance)  Bel\_LiDAR(Lane 2) = 0.4 (40% chance)  Bel\_LiDAR(Lane 3) = 0.3 (30% chance)  Bel\_LiDAR ({}) = 0 (Empty set, no opinion)  3. Dempster's Rule of Combination:  We need to combine Bel\_Camera and Bel\_LiDAR using Dempster's rule. This involves calculating the normalized weighting factor (denominator) and then multiplying the corresponding belief functions from each sensor for each combination of propositions.  Normalization Factor (W):  W = 1 - Sum(Bel\_Camera(A) \* Bel\_LiDAR(not A)) for all A in Theta except {}  W = 1 - (0.2 \* 0.7) + (0.6 \* 0.6) + (0.2 \* 0.7) = 0.2  Combined Belief Function (Bel):  Bel(Lane 1) = (Bel\_Camera(Lane 1) \* Bel\_LiDAR(Lane 1) + Bel\_Camera(Lane 2) \* Bel\_LiDAR(Lane 3)) / W  = (0.2 \* 0.3 + 0.6 \* 0.3) / 0.2 = 0.45  Bel(Lane 2) = (Bel\_Camera(Lane 1) \* Bel\_LiDAR(Lane 2) + Bel\_Camera(Lane 2) \* Bel\_LiDAR(Lane 2) + Bel\_Camera(Lane 3) \* Bel\_LiDAR(Lane 1)) / W  = (0.2 \* 0.4 + 0.6 \* 0.4 + 0.2 \* 0.3) / 0.2 = 0.4  Bel(Lane 3) = (Bel\_Camera(Lane 1) \* Bel\_LiDAR(Lane 3) + Bel\_Camera(Lane 2) \* Bel\_LiDAR(Lane 1) + Bel\_Camera(Lane 3) \* Bel\_LiDAR(Lane 3)) / W  = (0.2 \* 0.3 + 0.6 \* 0.3 + 0.2 \* 0.4) / 0.2 = 0.45  Bel({}) = 0 (Since no conflict arose during combination)  4. Analysis:  Basic Probability Assignments (BPA):  Bel(Lane 1) = 0.45  Bel(Lane 2) = 0.4  Bel(Lane 3) = 0.45  These values represent the car's belief in the object being in each lane based on the combined sensor data.  Plausibility Measure (Pl):  Pl(Lane 1) = 1 - Bel(Lane 2 U Lane 3) = 1 - (0.4 + 0.45) = 0.15 (represents maximum belief that the object could be in Lane 1)  Pl(Lane 2) = 1 - Bel(Lane 1 U Lane 3) = 1 - (0.45 + 0.45) = 0.1 (represents maximum belief that the object could be in Lane 2)  Pl(Lane 3) = 1 - Bel(Lane 1 U Lane 2) = 1 - (0.45 + 0.4) = 0.15 (represents maximum belief that the object could be in Lane 3) | 10 | 3 | 3 | 3 | 3.3.1 |
|  | In a medical environment, a Bayesian belief network (BBN) is utilized to help diagnose a rare but potentially life-threatening disease. The network considers symptoms A, B, and C, alongside test results, to estimate the likelihood of the patient having the disease.  How would you design a Bayesian belief network to model this scenario, considering the symptoms and test results? Additionally, how would you use the network to update the probability of the patient having the disease based on observed symptoms and test results?  **Answer: we need to consider the variables involved and their relationships. Let's denote the variables as follows:**  **Disease (D): Whether the patient has the rare but potentially life-threatening disease.**  **Symptom A (A): Presence or absence of symptom A.**  **Symptom B (B): Presence or absence of symptom B.**  **Symptom C (C): Presence or absence of symptom C.**  **Test Result (T): The result of a diagnostic test for the disease.**  **Here's how we can model the relationships between these variables:**  **Disease (D):**  **Prior probability of disease occurrence in the general population.**  **Symptoms (A, B, C):**  **The presence of symptoms may indicate the likelihood of the disease, but the absence of symptoms does not rule out the disease.**  **Conditional probabilities of symptoms given the presence or absence of the disease.**  **Test Result (T):**  **The diagnostic test's sensitivity and specificity are crucial here.**  **The test result is dependent on both the disease status and possibly other factors.**  **The Bayesian belief network structure would look like this:**  **\_\_\_\_\_ \_\_\_\_\_\_\_**  **/ \ / \**  **A --> D --> T --> B --> C**  **\\_\_\_\_\_/ \\_\_\_\_\_\_\_/**  **In this network:**  **Disease (D) is the parent node of Symptoms A, B, and C, as well as the Test Result (T).**  **Symptoms A, B, and C are conditionally dependent on the Disease node.**  **Test Result (T) depends on Disease (D), and Symptoms A, B, and C. It provides additional evidence for or against the presence of the disease.**  **To update the probability of the patient having the disease based on observed symptoms and test results, we use Bayesian inference. Given observed evidence (symptoms and test results), we can calculate the posterior probability of the disease using Bayes' theorem:**  **P(D∣E)=P(E∣D)⋅P(D)/ P(E)**  **​**  **Where:**  **P(D∣E) is the posterior probability of the disease given evidence E.**  **P(E∣D) is the likelihood of observing evidence E given the disease D.**  **P(D) is the prior probability of the disease.**  **P(E) is the marginal probability of observing evidence E, often calculated as the sum of**  **P(E∣D)⋅P(D) over all possible diseases.**  **In our case, evidence E includes the observed symptoms and test results. We can update the probability of the disease based on this evidence and then use the updated probabilities for decision-making, such as diagnosis and treatment planning.** | 10 | 3 | 3 | 3 | 3.3.1 |

**\*Performance Indicators are available separately for Computer Science and Engineering in AICTE examination reforms policy.**

**Course Outcome (CO) and Bloom’s level (BL) Coverage in Questions** 

**Approved by the Audit Professor/Course Coordinator**