VIT
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

REG.NO.:

SLOT: D1+TD1

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
CONTINUOUS ASSESSMENT TEST - I
FALL SEMESTER 2025-2026

Programme Name & Branch	: B.Tech. & Computer Science & Engineering / SCOPE
Course Code and Course Name	: Artificial Intelligence & BCSE306L
Faculty Name(s)	: All
Class Number(s)	: Common for all batches
Date of Examination	: 20-8-2025
Exam Duration	: 90 minutes

Maximum Marks: 50

General instruction(s):

- Answer All Questions

Q. No	Question	M
1.	An AI agent is supposed to manage the energy consumption of a smart home. The agent's objective is to minimize the electricity bill while ensuring the residents' comfort. It has access to weather forecasts, real-time electricity pricing from the grid, and sensors in each room (temperature, occupancy). It can control the heating, ventilation, and air conditioning (HVAC) system, the smart lighting, and high-consumption appliances like the dishwasher and washing machine. What is the correct PEAS categorization for this agent? Classify this task environment according to the following properties- Deterministic vs. Stochastic, Static vs. Dynamic, and Episodic vs. Sequential, and briefly justify your choices. What strategy should the agent adopt to effectively balance cost minimization with maintaining comfort?	10
2.	a) A Roomba vacuum cleaner's objective is a completely clean room. To achieve this, it uses its internal map to plan the most efficient path, ensuring it systematically covers every area without repetition. Which agent architecture best describes the Roomba? Explain with schematic diagram.	5
	b) What is the key difference between a Goal-Based Agent and a Utility-Based Agent? Provide an example where a Utility-Based Agent would be more suitable compared to a goal based agent.	5
3.	A Weighted Maze problem is having a 6x6 grid that represents a maze with different types of terrain as given below. Your goal is to find the path with the minimum cost from a starting point 'S' to a goal 'G'.	10

S	.	.	#	.	.
.	#	.	.	.	#
.	#	.	W	.	.
.	.	.	#	#	.
#	W	W	.	.	.
.	.	.	.	#	G

Maze Rules:

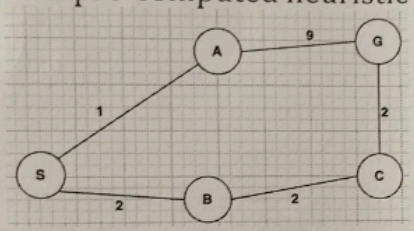
- You can move up, down, left, or right.
- You cannot move into cells, marked with '#'.
- Movement has a cost associated with the terrain type:

REG.NO.:

SLOT: D1+TD1

VIT
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
CONTINUOUS ASSESSMENT TEST - I
FALL SEMESTER 2025-2026

	○ Moving into a normal path cell ('.') has a cost of **1**. ○ Moving into a watery swamp cell ('W') has a cost of **2**. Represent systematic exploration of paths to find the optimal one using Uniform Cost Search algorithm. **Hint:** states can be represented as coordinate tuples (row, col), with the initial state at (0, 0) and the goal at (5, 5).	
4.	a) An autonomous delivery drone must navigate from a starting depot ('S') to a final destination ('G'). The drone can travel between several intermediate waypoints ('A', 'B', 'C'). The actual travel cost (e.g., energy consumed) between waypoints is known. To guide its search for the most efficient route, the drone's A* algorithm can use one of two different pre-computed heuristic functions, h1(n) or h2(n).  What is the final path and its total cost from A to G when A* is conducted using the two heuristic functions, h1(n) and h2(n).?	5
	b) Based on the results from the two searches performed above, did both heuristics lead to the optimal path? Which heuristic (h1 or h2) is admissible here and explain optimality based on the admissible feature.	5
5.	A search problem is given and it involves finding the global maximum of a function that has multiple local maxima. While executing, Hill Climbing stops at a local maximum far from the global maximum. However, Simulated Annealing starts from the same initial state but eventually reaches the global maximum. Write the algorithms and compare the fundamental search strategies of both. Explain in detail why Hill Climbing got stuck at the local maximum while Simulated Annealing succeeded. Propose a modification to Hill Climbing that might help it overcome local maxima, and explain how this change would affect its search behavior.	10

Node (n)	h1(n)	h2(n)
S	5	5
A	8	3
B	3	8
C	1	1
G	0	0

****

Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

REG.NO.:

SLOT:

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
CONTINUOUS ASSESSMENT TEST - I
FALL SEMESTER 2025-2026

AI- Answer key- D1 slot

Q. No	Question	M	CO	BL
1.	**PEAS Framework Analysis (4 marks)** **Performance Measure:** A complex trade-off between minimizing the total electricity cost and maximizing resident comfort. This could be measured by the monthly utility bill and a comfort score based on maintaining temperature within a preferred range and running appliances as needed. **Environment:** The physical house, the residents with their usage patterns, the external weather conditions, and the electricity grid with its variable pricing. **Actuators:** Controls for the HVAC system (turning on/off, adjusting temperature), smart light switches, and controls for smart appliances (e.g., starting a washing machine cycle). **Sensors:** Thermometers, occupancy sensors, data feeds for weather forecasts and real-time electricity prices, and sensors on appliances to know their status **Environment Classification (3 marks)** **Stochastic:** The environment is stochastic because the agent cannot perfectly predict future states. Resident behavior (e.g., opening a window), sudden weather changes not in the forecast, and fluctuations in electricity prices introduce elements of randomness. **Dynamic:** The environment is dynamic because it changes while the agent is deliberating. The temperature in the house changes, residents move from room to room, and electricity prices can update independently of the agent's actions. **Sequential:** The environment is sequential because current decisions have long-term consequences. For example, pre-cooling the house during a cheap electricity period will affect the energy needed later when the price is high. Actions are not independent episodes **Agent Architecture schematic diagram, Justification (3 marks)** The most suitable architecture is a **Utility-Based Agent**. **Justification:** A simple Goal-Based Agent might have the goal of "keeping the temperature at 22°C," but this is inefficient. It doesn't allow for nuanced trade-offs, such as letting the temperature drift slightly to avoid running the AC during peak electricity prices.	10		

2.	a) A Roomba's goal is a clean room; it plans a path to cover the entire area. This ability to plan based on an ultimate objective is what separates it from a simpler Model-Based *Reflex* agent, which would only use its map to react to its immediate surroundings. A Goal-Based agent architecture is the best description because its defining characteristic is the ability to use future outcomes to inform current decisions. The Roomba isn't just reacting to its environment; it has a specific, desirable future state—a "completely clean room"—and it actively plans a sequence of actions to reach that objective. **Diagram(2 marks) Explanation(3 marks)**	5		
	b) The key difference is that a Goal-Based agent works towards a simple, binary outcome (the goal is either achieved or not), while a Utility-Based agent seeks to maximize a continuous measure of "happiness" or "utility," allowing it to handle complex trade-offs and uncertainty. Goal-Based Agents operate on a "crude, binary distinction between 'happy' (goal achieved) and 'unhappy' (goal not achieved) states". This makes it difficult for them to choose between multiple paths that all lead to the same goal or to handle conflicting goals. Utility-Based Agents use a utility function, which assigns a numerical value to any given state to measure its desirability. This allows for a much more nuanced comparison of outcomes, enabling the agent to make rational decisions even when goals conflict or when no single goal can be achieved with certainty. **Example:** An automated taxi provides a clear example of where a Utility-Based agent is more suitable. A Goal-Based taxi has the simple goal: "get to the destination." Any route that achieves this is considered equally successful. Utility-Based taxi is superior because many different paths can reach the destination, but some are "better" than others. Its utility function would weigh multiple competing factors to find the *optimal* route, such as: o Travel time (speed) o Passenger comfort o Safety margins o Fuel efficiency and cost o Legality By maximizing a utility score that combines these preferences, the agent can make a more intelligent and high-quality decision than one that simply aims to complete a binary goal.	5		
3.	Step | Node Expanded | Explored Set | Frontier (Priority | Notes	10		

	(Cost, Coords)		Queue after expansion)	
1	Start	{}	[(0, (0,0))]	The search begins. The starting node (0,0) is added to the frontier with a cost of 0.
2	(0, (0,0))	{(0,0)}	[(1, (0,1)), (1, (1,0))]	Pop (0,0). Its neighbors (0,1) and (1,0) are added to the frontier, each with a cost of 1.
3	(1, (0,1))	{(0,0), (0,1)}	[(1, (1,0)), (2, (0,2))]	Pop (0,1). Its neighbor (0,2) is added with a cumulative cost of 2. (1,0) is now the cheapest node.
4	(1, (1,0))	{(0,0), (0,1), (1,0)}	[(2, (0,2)), (2, (2,0))]	Pop (1,0). Its neighbor (2,0) is added with a cost of 2. The frontier now has two paths with a cost of 2.
5	(2, (0,2))	{(0,0), (0,1), (1,0), (0,2)}	[(2, (2,0)), (3, (1,2))]	Pop (0,2). Its neighbor (1,2) is added with a cost of 3.
6	(2, (2,0))	{(0,0),...,(0, 2), (2,0)}	[(3, (1,2)), (3, (3,0))]	Pop (2,0). Its neighbor (3,0) is added with a cost of 3.
7	(3, (1,2))	{(0,0),...,(2, 0), (1,2)}	[(3, (3,0)), (4, (1,3)), (4, (2,2))]	Pop (1,2). Its neighbors (1,3) and (2,2) are added, each with a cumulative cost of 4.
8	(3, (3,0))	{(0,0),...,(1, 2), (3,0)}	[(4, (1,3)), (4, (2,2)), (4, (3,1))]	Pop (3,0). Its neighbor (3,1) is added with a cost of 4.
9	(4, (1,3))	{(0,0),...,(3, 0), (1,3)}	[(4, (2,2)), (4, (3,1)), (5, (1,4))]	Pop (1,3). Its neighbor (1,4) is added with a cost of 5.
10	(4, (2,2))	{(0,0),...,(1, 3), (2,2)}	[(4, (3,1)), (5, (1,4)), (5, (3,2)), (6, (2,3))]	Pop (2,2). Its neighbor (3,2) costs 1, so its path cost is 5. Its neighbor (2,3) is a swamp ('W') costing 2, so its path cost is 6.
11	(4, (3,1))	{(0,0),...,(2, 2), (3,1)}	[(5, (1,4)), (5, (3,2)), (6, (2,3)), (6, (4,1))]	Pop (3,1). Its neighbor (4,1) is a swamp ('W'), so its path cost is 6.
12	(5, (1,4))	{(0,0),...,(3, 1), (1,4)}	[(5, (3,2)), (6, (2,3)), (6,	Pop (1,4). Its neighbor (2,4) is added with a cost of 6.

				(4,1)), (6, (2,4))]				
	13	(5, (3,2))	{(0,0),...,(1, 4), (3,2)}	[(6, (2,3)), (6, (4,1)), (6, (2,4))]	Pop (3,2). Its neighbors are already explored or in the frontier with a better or equal cost.			
	14	(6, (2,4))	{(0,0),...,(3, 2), (2,3), (4,1), (2,4)}	[(7, (2,5)), (8, (4,2)), (8, (5,1))]	After popping several nodes with cost 6, UCS pops (6, (2,4)). Its neighbor (2,5) is added with a cost of 7.			
	15	(7, (2,5))	{(0,0),...,(2, 4), (2,5)}	[(8, (3,5)), (8, (4,2)), (8, (5,1))]	Pop (7, (2,5)). Its neighbor (3,5) is added with a cost of 8.			
	16	(8, (3,5))	{(0,0),...,(2, 5), (3,5)}	[(8, (4,2)), (8, (5,1)), (9, (4,5))]	Pop (8, (3,5)). Its neighbor (4,5) is added with a cost of 9.			
	17	(9, (4,5))	{(0,0),...,(5, 1), (4,5)}	[(9, (5,0)), (10, (4,3)), (10, (5,2)), (10, (5,5))]	After popping other nodes with cost 8, UCS pops (9, (4,5)). The goal node (5,5) is discovered and added to the frontier with a cost of 10.			
	18	(10, (5,5))	{(0,0),...,(5, 2), (5,5)}	[(11, (5,3))]	After popping all remaining nodes with costs 9 and 10, UCS finally pops the goal node (5,5). The search terminates.			

4. a) The A* algorithm uses the evaluation function `f(n) = g(n) + h(n)`, where `g(n)` is the actual cost from the start and `h(n)` is the estimated cost to the goal. `f(n) = g(n) + h1(n)` . The algorithm will expand the node with the lowest `f(n)` value from the frontier at each step. — 5

Step	Node Expanded	Frontier (Priority Queue	Explored Set	Path to Expanded Node
1	**Start**	``	`{}`	
2	**S**	``	`{S}`	S
3	**B**	`[(C, 5), (A, 9)]`	`{S, B}`	S → B
4	**C**	`[(G, 6), (A, 9)]`	`{S, B, C}`	S → B → C
5	**G**	`[(A, 9)]`	`{S, B, C, G}`	S → B → C → G

Result with Heuristic 1:

- **Nodes Expanded:** S, B, C, G
- **Final Path:** S → B → C → G
- **Total Cost: 6**

2. Trace with Heuristic 2 (`h2`) repeat the process with the second heuristic function.

Step	Node Expanded	Frontier (Priority Queue: [(	Explored Set	Path to Expanded Node
1	**Start**	``	{}	
2	**S**	``	{S}	S
3	**A**	``	{S, A}	S → A
4	**G**	``	{S, A, G}	S → A → G

(Note: In step 4, the algorithm expands G because its f-cost (10) is equal to B's, and in a tie-break, it may be chosen. Since G is the goal, the search terminates.)

Result with Heuristic 2:

- **Nodes Expanded:** S, A, G
- **Final Path:** S → A → G
- **Total Cost: 10**

b) Did both heuristics lead to the optimal path? No. Heuristic `h1` found the optimal path S → B → C → G with a cost of 6. Heuristic `h2` found the sub-optimal path S → A → G with a cost of 10.

5

- **Which heuristic is admissible?** An admissible heuristic never overestimates the true cost to the goal. Let's calculate the true costs (`h*(n)`) from each node to G:
  - `h*(S)` = 6 (via S→B→C→G)
  - `h*(A)` = 9 (via A→G)
  - `h*(B)` = 4 (via B→C→G)
  - `h*(C)` = 2 (via C→G)

  check our heuristics against these true costs:

  - **Heuristic 1 (`h1`) is admissible.** Every `h1(n)` value is less than or equal to the corresponding `h*(n)` value (e.g., `h1(B)=3 ≤ h*(B)=4`).
  - **Heuristic 2 (`h2`) is non-admissible.** It overestimates the cost from node B. The true cost from B to G is 4, but `h2(B)` gives an estimate of 8.

	The A* algorithm is only guaranteed to find the optimal path if its heuristic is admissible. The failure of `h2` can be traced to the specific moment it was used to evaluate node **B**. 1. **The Misleading Calculation:** When the algorithm expanded the start node 'S', it evaluated the paths to its neighbors 'A' and 'B': o **Path to A:** `f(A) = g(A) + h2(A) = 1 + 3 = 4` o **Path to B:** `f(B) = g(B) + h2(B) = 2 + 8 = 10` 2. **The Sub-Optimal Decision:** The non-admissible heuristic `h2(B) = 8` incorrectly made the path through B seem extremely expensive (`f(B) = 10`). In reality, the true cost from B is only 4, so a better estimate would have resulted in a much lower f-value. Because `f(A) = 4` was significantly lower, the algorithm was misled into believing that the path through A was far more promising			
5.	**Reason for Hill Climbing Failure:** **Algorithm(1.5 marks) Explanation(2 marks)** o Hill Climbing is a greedy local search algorithm. It moves only to neighbouring states with better values. o Once it reaches a state where no neighbour is better (a local maximum), it stops. o In this case, it got stuck at a local maximum because there was no mechanism to escape to a worse state that could lead to a better peak later. **Why Simulated Annealing Succeeded:** **Algorithm(1.5 marks) Explanation(2 marks)** o Simulated Annealing allows occasional moves to worse states, controlled by a probability that decreases over time (temperature parameter). o Early in the search (high temperature), the algorithm explores widely and can escape local maxima. o As the temperature decreases, it focuses more on exploitation, refining the solution near promising regions. o This combination of exploration (accepting worse solutions) and exploitation (gradually focusing on better ones) helped it reach the global maximum.	10		

Proposed Modification to Hill Climbing:(3 marks)

Variant	Description	Key Advantage	Limi
Stochastic Hill Climbing	Chooses a random neighbor from the current state, evaluates it, and moves there if it's better (rather than picking the best among all neighbors).	Faster in large search spaces as it doesn't evaluate all neighbors.	
First-Choice Hill Climbing	Randomly generates neighbors until it finds one better than the current state, then immediately moves to it.	Saves time by stopping neighbor checks early.	
Random-Restart Hill Climbing	Runs hill climbing multiple times from different random starting states and keeps the best solution found.	Can escape local maxima and increases chance of finding global optimum.	

****