USN | | | | | | | | | | |

Internal Assessment Test 3 – Jan 2022

Sub:	Introduction to Artificial Intelligence					Sub Code:	18CS753	Branch:	ECE/EEE/ME/CIV	
Date:	25/01/22	Duration:	90 mins	Max Marks:	50	Sem / Sec:	VII	OBE		
Answer any FIVE FULL Questions								MARKS	CO	RBT
1.	Outline the algorithm for minimax(position, depth, players) and explain							[10]	CO3	L2
2.	Explain Iterative deepening. Write algorithms for Depth-First iterative deepening and Iterative deepening- A*							[10]	CO3	L2
3.	Apply Candidate Elimination Algorithm to narrow the version space and show its trace using an example.							[10]	CO3	L3
4a.	Explain Knowledge acquisition process with example.							[05]	CO4	L2
4b.	Explain with example Learning by taking advice							[05]	CO4	L2
5a.	Explain Rote learning							[05]	CO4	L2
5b.	Make use of learning the concept CUP and write a note on Explanation- Based Learning							[05]	CO4	L3
6	What is learning. Explain Winston's learning program with example.							[10]	CO4	L1

USN | | | | | | | | | | |

Internal Assessment Test 3 – Jan 2022

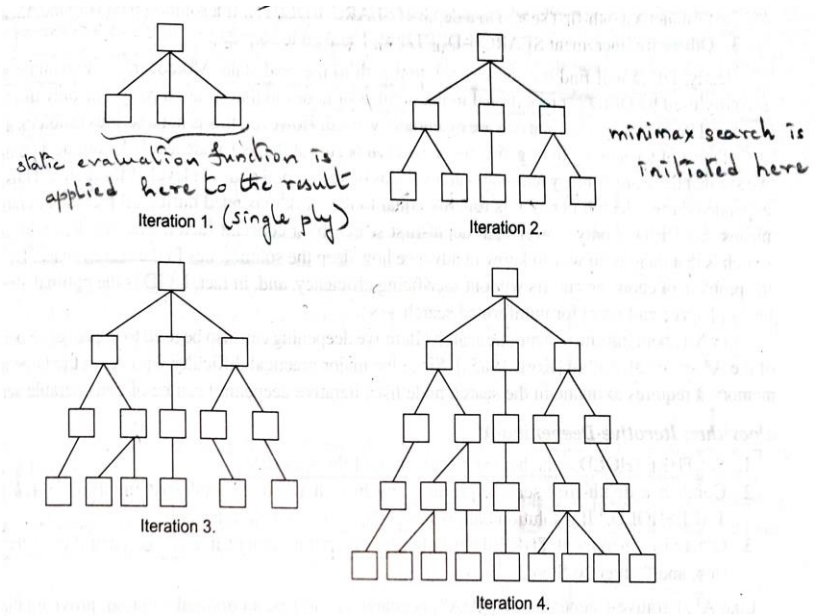
Sub:	Introduction to Artificial Intelligence					Sub Code:	18CS753	Branch:	ECE/EEE/ME/CIV	
Date:	25/01/22	Duration:	90 mins	Max Marks:	50	Sem / Sec:	VII	OBE		

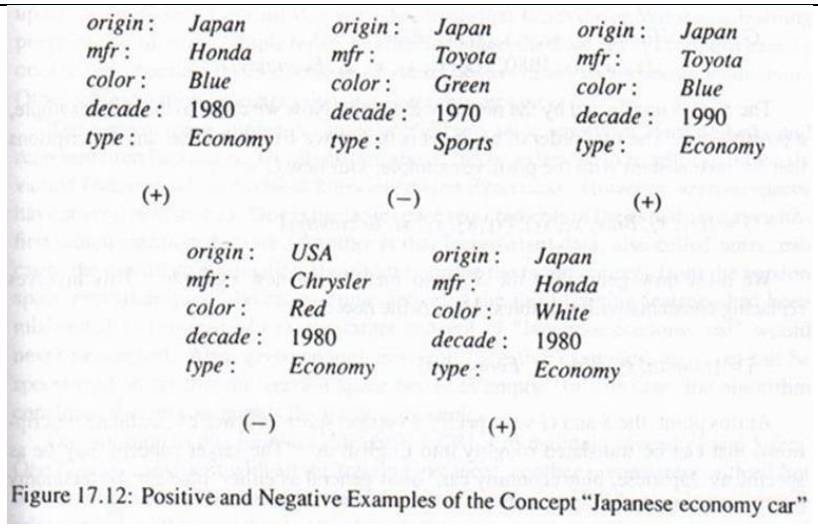
Answer any FIVE FULL Questions

		MARKS	CO	RBT

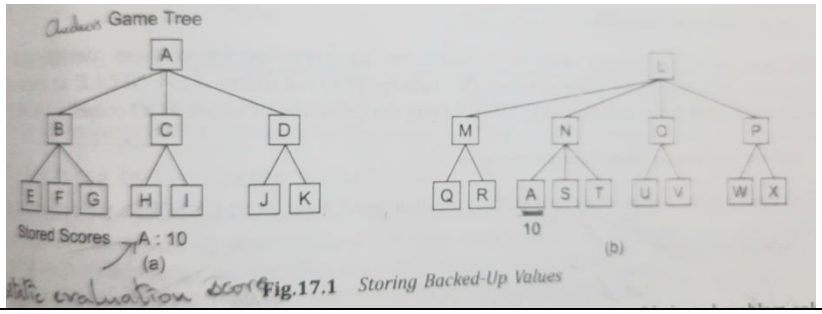
1.	Outline the algorithm for minimax(position, depth, players) and explain
	
	• MINIMAX needs to return 2 results:
	1. The backed-up value of the path it chooses.
	2. The path itself.
	• MINIMAX returns a structure containing both results & the two functions, VALUE and PATH extract separate components.
	• Initially it takes 3 parameters, a board position, the current depth of search and the player to move.
	• The initial call to compute best move from CURRENT position is
	
	`MINIMAX(CURRENT, 0, PLAYER-ONE)`
	if PLAYER-ONE is to move, or
	`MINIMAX(CURRENT, 0, PLAYER-TWO)`
	if PLAYER-TWO is to move.
	
	Algorithm: MINIMAX(Position, Depth, Player)
	1. If DEEP-ENOUGH(*Position, Depth*), then return the structure
	VALUE = *STATIC*(*Position, Player*);
	PATH = nil
	This indicates that there is no path from this node and that its value is that determined by the static evaluation function.
	2. Otherwise, generate one more ply of the tree by calling the function MOVE-GEN(Position Player) and setting SUCCESSORS to the list it returns.
	3. If SUCCESSORS is empty, then there are no moves to be made, so return the same structure that would have been returned if DEEP-ENOUGH had returned true.
	4. If SUCCESSORS is not empty, then examine each element in turn and keep track of the best one. This is done as follows.
	Initialize BEST-SCORE to the minimum value that STATIC can return. It will be updated to reflect the best score that can be achieved by an element of SUCCESSORS.
	For each element SUCC of SUCCESSORS, do the following:
	(a) Set RESULT-SUCC to
	MINIMAX(SUCC, *Depth* + 1, OPPOSITE(*Player*)) } recursive call.
	This recursive call to MINIMAX will actually carry out the exploration of SUCC.
	(b) Set NEW-VALUE to - VALUE(RESULT-SUCC). This will cause it to reflect the merits of the position from the opposite perspective from that of the next lower level.
	(c) If NEW-VALUE > BEST-SCORE, then we have found a successor that is better than any that have been examined so far. Record this by doing the following:
	(i) Set BEST-SCORE to NEW-VALUE.
	(ii) The best known path is now from CURRENT to SUCC and then on to the appropriate path down from SUCC as determined by the recursive call to MINIMAX. So set BEST-PATH to the result of attaching SUCC to the front of PATH(RESULT-SUCC).
	5. Now that all the successors have been examined, we know the value of Position as well as which path to take from it. So return the structure
	VALUE = BEST-SCORE
	PATH = BEST-PATH
	When the initial call to MINIMAX returns, the best move from CURRENT is the first element on PATH
		[10]	CO3	L2

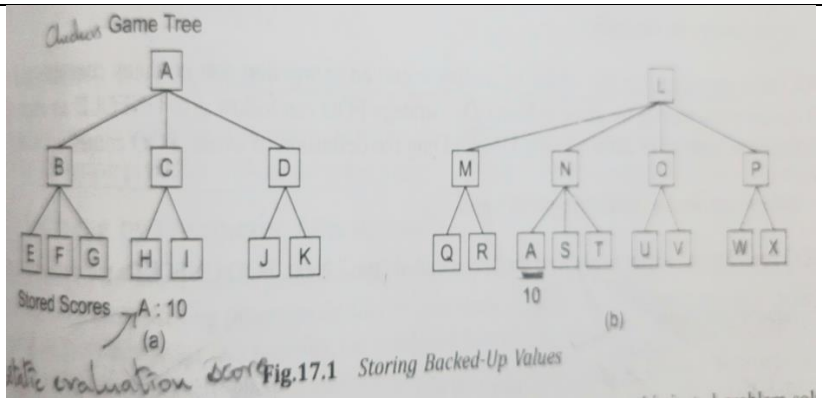
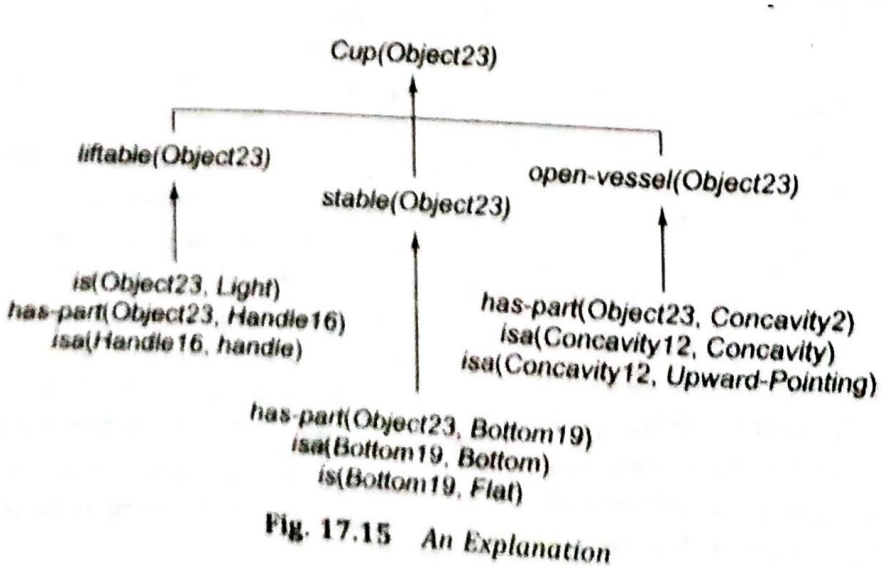
2.	Explain Iterative deepening. Write algorithms for Depth-First iterative deepening and Iterative deepening- A* • The name "***iterative deepening***" derives from the fact that on each iteration,	[10]	CO3	L2

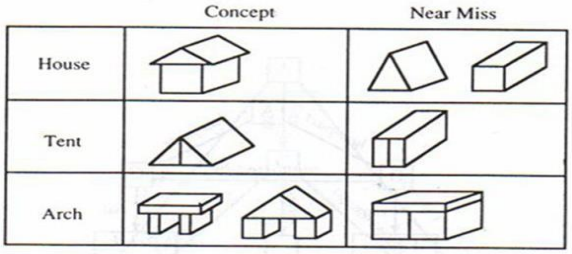
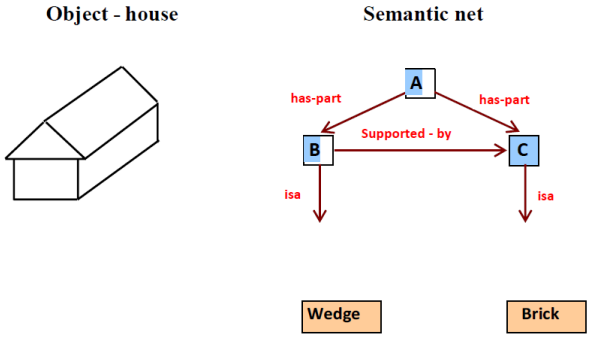
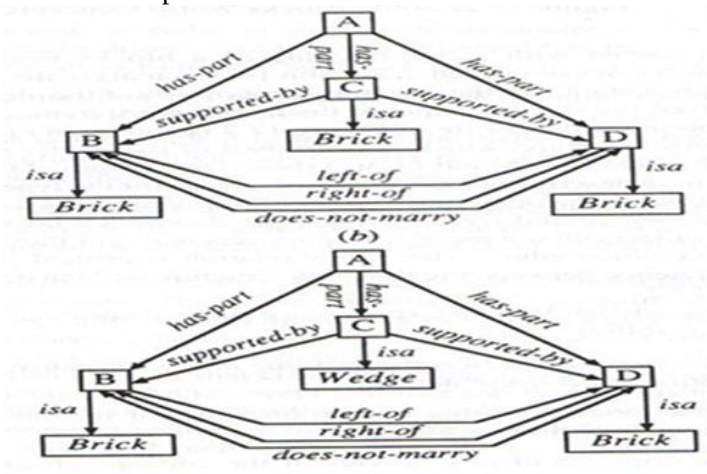
	the tree is searched one level deeper.  **Fig. 12.10** *Iterative Deepening* • An algorithm called depth-first iterative deepening(DFID) combines the best aspects of depth-first & breadth-first search. ***Algorithm: Depth-First Iterative Deepening*** 1. Set SEARCH-DEPTH = 1. 2. Conduct a depth-first search to a depth of SEARCH-DEPTH. If a solution path is found, then return it. 3. Otherwise, increment SEARCH-DEPTH by 1 and go to step 2. • Clearly, DFID will find shortest solution path to goal state. • The maximum amount of memory used by DFID is proportional to number of nodes in that solution path. • DFID is optimal algorithm in terms of space and time for uninformed search. • Iterative deepening can also be used to improve the performance of A* search algorithm. • The major practical difficulty with A* is the large amount of memory it requires to maintain the search node lists. ***Algorithm: Iterative-Deepening-A**** 1. Set THRESHOLD = the heuristic evaluation of the start state. 2. Conduct a depth-first search, pruning any branch when its total cost function ($g + h'$) exceeds THRESHOLD.[4] If a solution path is found during the search, return it. 3. Otherwise, increment THRESHOLD by the minimum amount it was exceeded during the previous step, and then go to Step 2. • Iterative- Deepening A* is guaranteed to find an optimal solution, provided that h' is an admissible heuristic.			
3.	Apply Candidate Elimination Algorithm to narrow the version space and show its trace using an example. • The algorithm for narrowing VS is called Candidate Elimination Algorithm. • Tracing concept of CEA for the concept of ***"Japanese Economy Car"***	[10]	CO3	L3

	 Figure 17.12: Positive and Negative Examples of the Concept "Japanese economy car" • G contains NULL and S contains 1st positive training example. $G = \{(x_1, x_2, x_3, x_4, x_5)\}$ $S = \{(Japan, Honda, Blue, 1980, Economy)\}$ • G is specialized to eliminate –ve example from VS $G = \{(x_1, Honda, x_3, x_4, x_5), (x_1, x_2, Blue, x_4, x_5), (x_1, x_2, x_3, 1980, x_5), (x_1, x_2, x_3, x_4, Economy)\}$ • Remove values that are inconsistent with +ve example • Replace constants with variables $G = \{(x_1, x_2, Blue, x_4, x_5), (x_1, x_2, x_3, x_4, Economy)\}$ $S = \{(Japan, x_2, Blue, x_4, Economy)\}$ • S and G sets specify a version space. • S is unaffected but generalizing G set w.r.t –ve training example. $G = \{(Japan, x_2, Blue, x_4, x_5), (Japan, x_2, x_3, x_4, Economy)\}$ • Remove form G , inconsistent descriptions w.r.t +ve training example. $G = \{(Japan, x_2, x_3, x_4, Economy)\}$ $S = \{(Japan, x_2, x_3, x_4, Economy)\}$ • S and G both are singleton. • Algorithm has converged on Target Concept.			
4a.	Explain Knowledge acquisition process with example. • Typically, a knowledge engineer interviews a domain expert to Elucidate expert knowledge, which is then translated into rules. • After the initial system is built, it must be iteratively refined until it approximates expert-level performance. • There are many programs that interact with domain experts to extract expert knowledge efficiently. • These programs provide support for the following activities: 1. Entering knowledge. 2. Maintaining knowledge base consistency. 3. Ensuring knowledge base completeness. • 2 knowledge acquisition systems are as follows:	[05]	CO4	L2

	1. MOLE and 2. SALT • MOLE is knowledge acquisition system for heuristic classification problems, such as diagnosing diseases. • An expert systems produced by MOLE accepts input data, comes up with a set of candidate explanations or classifications that cover the data, then uses differentiating knowledge to determine which one is best. • MOLE interacts with domain expert to produce a knowledge base that a system called MOLE-p uses to solve problems. • The acquisition proceeds through several steps: 1. Initial knowledge base construction : MOLE tries to determine the conditiond under which one explanation is correct. • The expert provides covering knowledge i.e., the knowledge that a hypothesized event might be the cause of certain symptom. • MOLE then tries to infer anticipatory knowledge. 2. Refinement of knowledge base: MOLE now tries to identify the weakness of knowledge base. • One approach is to find holes & prompt the expert to fill them. • Whenever MOLE-p makes an incorrect diagnosis, the expert adds new knowledge. • MOLE has been used to build systems that diagnose problems with car engines, problems in steel-rolling mills and inefficiencies in coal-burning power plants.			
4b.	Explain with example Learning by taking advice • When a programmer writes a series of instructions into a computer. • The programmer is a sort of teacher & the computer is a sort of student. • After being programmed, the computer is able to do something it previously could not. • Suppose the program is written in a high-level language like LISP. • Some interpreter or compiler must intervene to change the teacher's instructions into code that the machine can execute directly. • Mostow, describes a program called FOO, which accepts advice for playing hearts, a card game. • A human user first translates the advice from English into a representation that FOO can understand. • For example, " Avoid taking points" becomes: (avoid(take-points me) (trick)) • FOO must *operationalize* this advice by turning it into an expression that contains concepts & actions FOO can use when playing the game of hearts. • One strategy is: UNFOLD an expression by replacing some term by its definition. • FOO comes up with: (achieve(not(during(trick)(take-points me)))) • FOO considers advice to apply to the player called "me". • Next, FOO UNFOLDs definition of trick (achieve(not(during(scenario (each pl(players)(play-card pl) (take-trick (trick-winner))) (take-points me)))) • In other words , player should avoid taking points during the scenario consisting of	[05]	CO4	L2

	1. Players playing cards & 2. One player taking the trick • FOO then uses *case analysis* to determine which steps could cause one to take points. • Step 2 could affect taking points , so FOO-UNFOLDs the definition of take-points: (achieve (not (there-exists cl(cards-played) (there-exists c2 (point-cards) (during(take(trick-winner)cl) (take me c2)))))) • This device says that the player should avoid taking point-cards during the process of trick-winner taking the trick. • The question for FOO now is: Under what conditions does (take me c2) occur during (take(trick-winner)cl)? • By using technique called *partial match*, FOO hypothesizes that points will be taken if me= trick-winner and c2=c1. • It transforms the advice into: (achieve (not (and(have- points (cards-played)) (=(trick-winner) me)))) • This means " Do not win a trick that has points". • Through a number of other transformations, FOO eventually settles on: (achieve (>= (and(in-suit-led(card-of me)) (possible (trick-has-points))) (low(card-of me))) • At last, FOO has translated the rather vague advice "avoid taking points" into a specific, usable heuristic. • FOO is able to play a better game of hearts after receiving this advice.			
5a.	Explain Rote learning • It is mechanism of Caching • When computer stores piece of data, data is cached so that recomputing is not required. • Caching is used by AI to improve performance & such caching is known as *Rote Learning*. • *Samuel's Checkers* program used 2 types of learning: rote learning & parameter adjustment. • Samuels program used checkers game trees for representing states. • Time constraints permitted it to search only a few levels in tree. • Later static evaluation method was applied to board position. • Then used the score obtained from this function to search the game tree. • When search was complete, it propagated the score backwards and root position had a score. • It now chose the best move and also recorded the board position and score at tree root.  **Fig.17.1** *Storing Backed-Up Values*	[05]	CO4	L2

	 Fig.17.1 *Storing Backed-Up Values* • Instead of using the static evaluation function to compute a score for position A, the stored value for A can be used. • Rote learning of this sort is very simple. • But it needs some capabilities. • These capabilities include: ✓ *Organized Storage of Information*- in order for it to be faster to use a stored value than it would be to recompute it, there must be a way to access the appropriate stored value quickly. ✓ *Generalization*- to keep the number of stored objects to a manageable level, some kind of generalization is necessary.			
5b.	Make use of learning the concept CUP and write a note on Explanation- Based Learning • EBL programs accept the following as input: 1. **A Training Example:** What the Learning Program sees in the world. Ex: the Car 2. **A Goal Concept:** A high level description of what the program is supposed to learn 3. **An Operationally Criterion:** A description of which concepts are usable. 4. **A Domain Theory:** A set of rules that describe relationships between objects & actions in a domain. • From this EBL computes generalization of the training example that is sufficient to describe the goal concept & also satisfies the operationality criterion. • An EBL program seeks to operationalize the goal concept by expressing it in terms that a problem-solving program can understand.  **Fig. 17.15** *An Explanation*	[05]	CO4	L3

6	What is learning. Explain Winston's learning program with example. • **Learning** means: ***… changes in the system that are adaptive*** *i.e.. They enable s/y to do same task* *Or* *They enable s/y to do tasks more efficiently & effectively next time* • Learning covers wide range of phenomena and spectrum. • The goal was " to construct representations of the definitions of concepts in blocks domain. • Ex: it learned the concepts House, Tent and Arch shown in figure. • Near miss is also shown in figure. • **Near Miss:** its an *Object* similar to *instances* of *concept* in question.  Figure 17.2: Some Blocks World Concepts • The program started with a line drawing of a blocks world structure. • Then structural description was provided as input to learning program. • An example of such a structural description for the house is shown here. • Node A represents entire structure. Its composed of node B(Wedge) and C(Brick)  • Then structural description of arcs.  	[10]	CO4	L1

2 objects Marry: if they have faces that touch & have common edge.
Marry Relation= Arch – Near miss Arch

- In comparison of 2 arches: the objects represented by node C are not identical.
- **C-note** link describes **difference** found by **comparison routine.** The difference occurred in **isa link**.

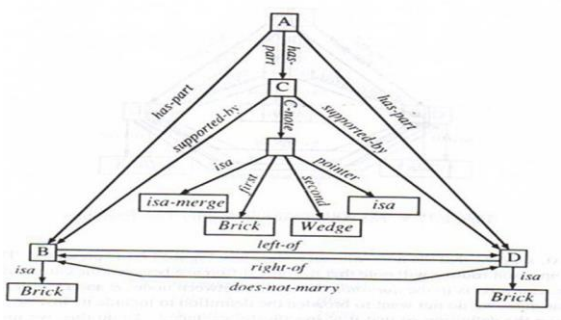


Figure 17.4: The Comparison of Two Arches

- At this point, new description of Arch is generated.
- Node C is either a Brick or a Wedge.
- At node Object: Brick & Wedge merge & arch is built as shown in figure here

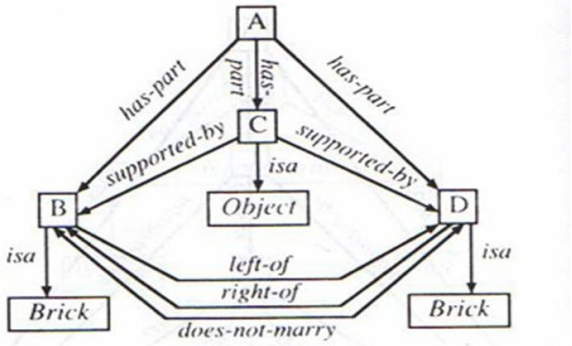


Figure 17.5: The Arch Description after Two Examples