T-F Questions:

Agents (Chapter 2):

- A rational agent acts so as minimize to the expected value of the performance measure, given the percept sequence it has seen so far F
- Simple reflex agent will take decisions correctly depending on the current percept. F
- Simple-reflex agents choose actions that optimize the utility function. F
- It is enough for goal-based agent to know about the current state of the environment. F
- We usually take into consideration omniscience when implementing agents. F
- Agents should act with omniscience. F
- Semi-dynamic refers to agent actions, not to the environment itself. T
- Rationality Means doing the right thing and taking the best action depending on the knowledge you have T
- Goals alone are not enough to generate high-quality behaviour, So Simple reflex agent is used to try to maximize their own "happiness." F
- Observable environment is more convenient than partial environments. T
- Intelligent Agent must be measurable as agent have to perform desirable actions T
- In a learning agent, the critic gives the learning element feedback on how the agent is doing good or bad. T
- A rational agent should be autonomous T
- A randomized simple reflex agent might outperform a deterministic one? T
- As a general rule it is better to design performance measures according to what one actually wants in the environment, rather than according to how one thinks the agent should behave. T
- There exists a task environment in which every agent is rational. T
- The input to an agent program is the same as the input to the agent function. F
- An agent that senses only partial information about the state cannot be perfectly rational. F
- There exist task environments in which no pure reflex agent can behave rationally. F
- Every agent function is implementable by some program/machine combination F

- Suppose an agent selects its action uniformly at random from the set of possible actions, there exists a deterministic task environment where this agent is rational T
- It is possible for a given agent to be perfectly rational in two distinct task environments T
- Every agent is rational in an unobservable environment F

Search (Chapter 3):

- Before an agent can start searching for solutions, a goal must be identified, and a well-defined problem must be formulated. T
- We can never reach goal state if there is no path between it and the current state. F
- Iterative deepening search is most useful when the depth of the solution is known in advance. F
- The execution phase is the last phase to design a problem-solving agent T
- DFS is used over BFS when you have limited memory T
- DFS always expands one of the nodes at the deepest level of the tree. T
- Space complexity and time complexity is the evaluation criteria for search strategies only F
- You Can Compare between AI Algorithms using Time Complexity Only F
- In alpha-beta pruning algorithm, alpha is the value of the best (i.e., highest-value) choice we have found so far at any choice point along the path for MAX ? T
- BFS may actually be faster than DFS. T
- The time complexity of depth-first tree search is bounded by the size of the state space F
- State space is composed of all the states generated by all possible available actions applied to the initial state as well as to any state generated. T
- Uninformed search uses problem-specific knowledge beyond the definition of the problem itself F
- An open-loop system is an agent that carries out its plans while paying attention to its percepts. F
- In Sokoban, the distance between the player and the farthest crate (box) is an admissible heuristic function. F
- A* Tree Search is incomplete if the state space graph contains cycles F
- Iterative deepening is complete and optimal if all the actions have the same cost. T
- Search Algorithms use factored state representation F

- in UCS, the first path from S to G to be inserted in the frontier is always the optimal path F
- Greedy search is always optimal and complete. F
- UCS graph search always gives the optimal solution. T
- It is possible that UCS generates a different path if we add a positive constant cost to each arc T
- Average of 2 admissible heuristics is also admissible T

Local Search (Chapter 4):

- A genetic algorithm is a stochastic hill-climbing search in which a large population of states are maintained. New states are generated by mutation and by crossover, which combines pairs of states from the population. T
- The new states are generated in genetic algorithm only through cross over. F
- Local search algorithms aim to find the path to the goal, but with reduced memory footprint. F
- Hill climbing does not look ahead beyond the immediate neighbours of the current state. T
- Local beam is used to solve memory problem for A* search T
- In nondeterministic environments, agents can apply AND –XOR search to generate contingent plans that reach the goal regardless of which outcomes occur during execution. F
- Local search algorithms are not systematic, and they have two key advantages; one of them is that they use very little memory—usually a constant amount T
- Local maximum is a peak that is higher than each of its neighbouring states but lower than the global maximum. T
- Increasing the sideways moves number in hill-climbing search algorithm can increase the percentage of the problem instances that can be solved. T
- A hill-climbing search might get lost on the plateau. T
- In Simulated Annealing, bad moves are likely to be allowed at the start when T is low, and they become more likely as T increases F
- Greedy hill climbing is complete if the search space is finite. F
- The space complexity of simulated annealing is $O(b^d)$ where b is the branching factor and d is the minimum number of steps needed to reach the global maximum starting from the initial state. F
- Hill climbing that always takes better moves will eventually get a solution if given enough time F
- In Local Search, if we move to the next node only if it has larger value, this will always guarantee an optimal solution after infinite iterations. F

- In simulated annealing when we set T to infinity, it is the same as stochastic hill climbing. F
- Genetic algorithms are a variant of hill climbing that combine two parent nodes to produce a new child node instead of generating them from a single state. F

Games (Chapter 5):

- The alpha–beta search algorithm computes the same optimal move as minimax but achieves much greater efficiency by eliminating subtrees that are probably irrelevant. T
- Collaboration can happen between 2 players in a non-zero-sum game. T
- We always need to examine each node in the tree to compute the correct minimax decision. F
- Alpha beta pruning is used to decrease the depth of the tree generated from minimax algorithm T
- Many game programs precompute tables of best moves in the opening and endgame so that they can look up a move rather than search. T
- Zero sum games could be a single player game T
- The problem with the minimax search is that the number of game states it has to examine is exponential in the depth of the tree. T
- The effectiveness of alpha-beta pruning is highly dependent on the order in which the states are examined. T
- In MiniMax alpha-beta pruning, the children to the first leftmost parent will never be pruned. T

CSP (Chapter 6):

- Local search using the min-conflicts heuristic is applied to constraint satisfaction problems with great success. T
- Breadth-first search is commonly used for solving CSPs. F
- 3-consistency is the same as arc consistency. F
- Violation of a preference constraint in a constraint satisfaction problem out a potential solution. F
- In the constraint graph arcs show constraints T
- The main idea of CSP is to eliminate large portions of the search space all at once by identifying variable/value combinations that violate the constraints T
- Many important real-world problems can be described as CSPs. T
- A variable in a CSP is arc-consistent if every value in its domain satisfies the variable's binary constraints. T

- Backtracking Search is a form of Breadth First Search used for solving constraint satisfaction problems (CSPs) F
- The complexity of solving a constraint satisfaction problem is not related to the structure of its constraint graph. F
- In CSPs, minimum-remaining-values is a domain-independent
- method for deciding which variable to choose next T
- The values that are left in the domain after applying arc consistency depend on the order of processing the arcs in the queue F
- By using the most-constrained variable heuristic and the least-constraining value heuristic we can solve every CSP in time linear in the number of variables. F
- In CSP, if it is arc consistent, it can be solved without backtracking. F
- Consider a CSP in a tree structure and arc consistency was ensured on all arcs and we assign variables from the root to the leaf. Does this guarantee no backtracks? T

MDP (Chapter 17):

- If two MDPs are exact replicas except for the discount factor, then they will always have the same optimal policy F
- MDP deals with deterministic, partially observable, sequential environments F
- When we set the discount factor (gamma) to a value near zero, the resulting MDP will make the agent more greedy. T

Reinforcement Learning (Chapter 21):

- Temporal Difference Learning does not learn the transition model T
- In Q-Learning, in order to be optimal it has to take its Q-values from optimal policy. F