1. These questions address the topic of Intelligent Agents.

(a) 5 Points
True or False: To be an ideal agent, an agent program must explicitly contain a performance measure.

(b) 5 Points
True or False: Goal-based agents must be able to reason about the relative desirability of multiple goal states.

(c) 5 Points
An environment in which the next state is completely determined by the current state and the actions of the agent is said to be:
   a. Exciting
   b. Static
   c. Discrete
   d. Deterministic

(d) 5 Points
Which of the following kinds of agents reason about the results of their actions?
   a. Table-based
   b. Reflex-based
   c. Goal-based
   d. All of the above
2. The following questions address the topic of search.

(a) **5 Points**
What does it mean for a search algorithm to be optimal?

(b) **5 Points**
What does it mean for a search algorithm to be complete?

(c) **5 Points**
What exactly is it that uninformed search algorithms are uninformed about?

(d) **5 Points**
True or False: Breadth-First Search is never optimal.
(e) **5 Points**
True or False: A search algorithm that uses a heuristic can not be optimal.

(f) **5 Points**
True or False: A* search keeps its queue sorted based on the value of its heuristic function (h).
(g) **20 Points**

Identify the flaw that I’ve introduced into the DFS-Contour function in the following IDA* code and explain why it is a flaw.

```plaintext
function IDA*(problem) returns a solution sequence
    inputs: problem, a problem
    local variables: f-limit, the current f-Cost limit
        root, a node

    root ← MAKE-NODE(INITIAL-STATE[problem])
    f-limit ← f-Cost[root]
    loop do
        solution, f-limit ← DFS-Contour(root, f-limit)
        if solution is non-null then return solution
        if f-limit = ∞ then return failure; end

function DFS-Contour(node, f-limit) returns a solution sequence and
    a new f-Cost limit
    inputs: node, a node
        f-limit, the current f-Cost limit
    local variables: next-f, the f-Cost limit for the next contour, initially ∞

    if Goal-Test[problem](State[node]) then return node, f-limit
    if f-Cost[node] > f-limit then return null, f-Cost[node]
    for each node s in Successors(node) do
        solution, new-f ← DFS-Contour(s, f-limit)
        if solution is non-null then return solution, f-limit
        next-f ← MIN(next-f, new-f); end
    return null, next-f
```

5
3. The following questions address the topic of game playing.

(a) **5 Points**
A full 4-ply MiniMax search in a game with a branching factor of 3 will apply its evaluation function to how many boards?

- a. 64
- b. 81
- c. 121
- d. 120

(b) **5 Points**
True or False: A MiniMax search with Alpha-Beta pruning finds better moves than the same search without pruning.

(c) Alpha-Beta pruning is most effective when informative moves appear early in the search process.

i. **5 Points**
What exactly does “early” mean to in a typical MiniMax implementation?

ii. **15 Points**
Suggest a technique that has some promise of ensuring that informative moves appear early in the search.