1. For each of the following assertions, say whether it is true or false and support your answer with examples or counterexamples where appropriate:

a. An agent that senses only partial information about the state cannot be perfectly rational.

False. The vacuum-cleaning agent is rational but doesn't observe the state of the square that is adjacent to it,

b. There exist task environments in which no pure reflex agent can behave rationally.

True. The card game Concentration or Memory is one. Anything where memory is required to do well will thwart a reflex agent.

c. There exists a task environment in which every agent is rational.

True. Consider a task environment in which all actions (including no action) give the same, equal reward

d. The input to an agent program is the same as the input to the agent function.

False. The input to a agent function is the percept history. The input to a agent program is only the current percept; it is up to the agent program to record any relevant history needed to make actions.

e. Every agent function is implementable by some program/machine combination.

False. Consider an agent whose only action is to return an integer, and who perceives a bit each turn. It gains a point of performance if the integer returned matches the value of the entire bitstring perceived so far. Eventually, any agent program will fail because it will run out of memory.

f. Suppose an agent selects its action uniformly at random from the set of possible actions. There exists a deterministic task environment in which this agent is rational.

True. Again consider the "all actions always give equal reward" case

g. It is possible for a given agent to be perfectly rational in two distinct task environments.

True. Consider two environments based on betting on the outcomes of a roll of two dice. In one environment, the dice are fair, in the other, the dice are biased to always give 3 and 4. The agent can bet on what the sum of the dice will be, with equal reward on all possible outcomes for guessing correctly. The agent that always bets on 7 will be rational in both cases.

h. Every agent is rational in an unobservable environment.

False. Built-in knowledge can give a rational agent in an unobservable environment. A vacuum-agent that cleans, moves, cleans moves would be rational, but one that never moves would not be.

2. For each of the following activities, give a PEAS description of the task environment:

   a. **Playing soccer.**

      - P -Win/Lose | E- Soccer field | A- Legs, Head, Upper body | S- Eyes, Ears. Partially observable, multi-agent, stochastic, sequential, dynamic, continuous, unknown

   b. Shopping for used AI books on the Internet

      - P- Cost of book, quality/relevance/correct edition | E- Internet's used book shops | A- key entry, cursor | S- website interfaces, browser. partially observable, multiagent, stochastic, sequential, dynamic, continuous, unknown.

   c. **Playing a tennis match.**

      - P- Win/Lose | E- Tennis court | A- Tennis racquet, Legs | S- Eyes, Ears. partially observable, multi-agent, stochastic, sequential, dynamic, continuous, unknown.

   d. **Practicing tennis against a wall.**

      - P- Improved performance in future tennis matches | E- Near a wall | A- Tennis racquet, Legs | S- Eyes, Ears. observable, single agent, stochastic, sequential, dynamic, continuous, unknown.

   e. **Performing a high jump.**

      - P- Clearing the jump or not | E- Track | A- Legs, Body | S- Eyes. observable, single agent, stochastic, sequential, dynamic, continuous, unknown

3. Explore the differences between agent functions and agent programs.

   a. Can there be more than one agent program that implements a given agent function?

      - Yes. Assume we are given an agent function whose actions only depend on the previous p percepts. One program can remember the previous p percepts to implement the agent function, while another could remember greater than p percepts and still implement the same agent function. <u>[This includes the answer to b also]</u>

   b.

   c. Are there agent functions that cannot be implemented by any agent program? Yes.

   d. Given a fixed machine architecture, does each agent program implement exactly one agent function?

      - Yes. Given a percept sequence, an agent program will select an action. To implement multiple agent functions this would require the agent program to select different actions (or different distributions of actions) given the same percept sequence.

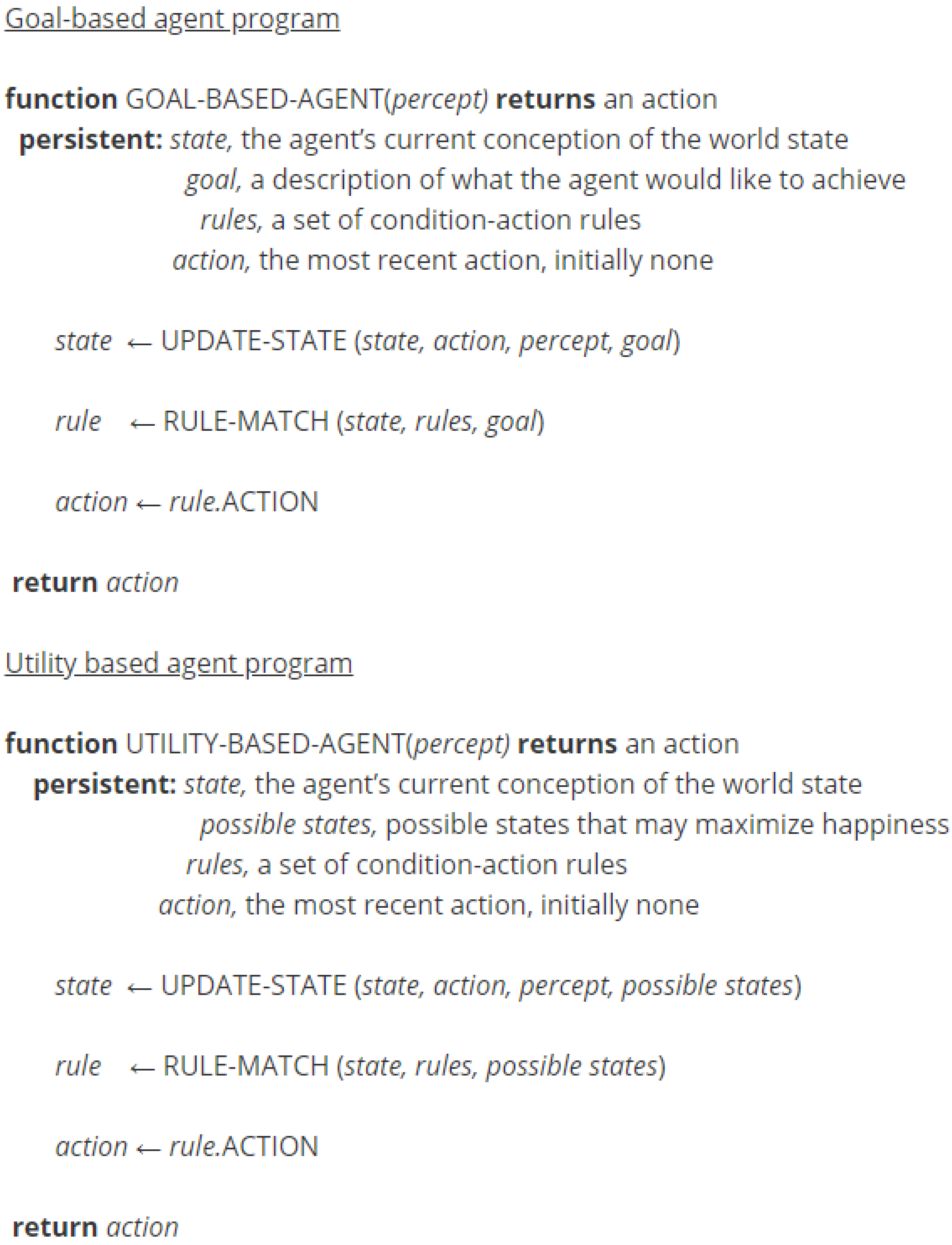
e. Given an architecture with n bits of storage, how many different possible agent programs are there?

- If a is the total number of actions, then the number of possible programs are a^{2n}, 2^n internal states and a choices for each state.

f. **Suppose we keep the agent program fixed but speed up the machine by a factor of two. Does that change the agent function?**

- No, not directly. However this may allow the program to compress it's memory further and to retain a better model of the world.

4. **Write pseudocode agent programs for the goal-based and utility-based agents**.



Goal-based agent program

```
function GOAL-BASED-AGENT(percept) returns an action
  persistent: state, the agent’s current conception of the world state
                    goal, a description of what the agent would like to achieve
                      rules, a set of condition-action rules
                  action, the most recent action, initially none

      state ← UPDATE-STATE (state, action, percept, goal)

      rule   ← RULE-MATCH (state, rules, goal)

      action ← rule.ACTION

 return action
```

Utility based agent program

```
function UTILITY-BASED-AGENT(percept) returns an action
   persistent: state, the agent’s current conception of the world state
                      possible states, possible states that may maximize happiness
                    rules, a set of condition-action rules
                 action, the most recent action, initially none

      state ← UPDATE-STATE (state, action, percept, possible states)

      rule   ← RULE-MATCH (state, rules, possible states)

      action ← rule.ACTION

 return action
```