1. (Exercise 2.3 from RN) 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 from Section 2.2.1 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.*

   (i) A perfectly playing poker-playing agent never loses. *False. Pit two perfectly playing agents against each other. Someone (the one with poorer luck) must lose.*

2. (Exercise 2.4) For each of the following activities, give a PEAS description of the task environment and characterize it in terms of the properties listed in Section 2.3.2 (Properties of Task Environments in RN 2nd ed.)

   - Playing soccer. *P- Win/Lose E- Soccer field A- Legs,Head,Upper body S- Eyes,Ears. partially observable, multiagent, stochastic, sequential, dynamic, continuous, unknown*

- Exploring the subsurface oceans of Titan. *P- Surface area mapped, extraterrestrial life found E- subsurface oceans of Titan A- steering, accelerator, break, probe arm, S- camera, sonar, probe sensors. partially observable, single agent, stochastic, sequential, dynamic, continuous, unknown*
- 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*
- Playing a tennis match. *P- Win/Lose E- Tennis court A- Tennis racquet, Legs S- Eyes, Ears. partially observable, multiagent, stochastic, sequential, dynamic, continuous, unknown*
- 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*
- 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*
- Knitting a sweater. *P- Quality of resulting sweater E- Rocking chair A- Hands,Needles S- Eyes. observable, single agent, stochastic, sequential, dynamic, continuous, unknown*
- Bidding on an item at an auction. *P- Item acquired, Final price paid for item E- Auction House (or online) A- Bidding S- Eyes, Ears. Partially observable, multiagent, stochastic (tie-breaking for two simultaneous bids), episodic, dynamic, continuous, known*

3. (Exercise 2.5) Define in your own words the following terms: agent, agent function, agent program, rationality, autonomy, reflex agent, model-based agent, goal-based agent, utility-based agent, learning agent. *Agent: An algorithmic entity capable of displaying intelligent-like behavior. Agent function: a mapping from input-sequences to actions defining the behavior of an agent. Agent program: physical program implementing or approximating an agent function. Rationality: the behavior of maximizing one's own reward or performance. Reflex agent: agent only capable of considering it's current perception of the world. Model-based agent: agent that attempt to internalize aspects of the world through an approximating model. Goal-based agent: agent whose performance measure does not directly depend on local actions but on some (potentially) distant goal. Utility-based agent: agent whose performance measure is given by a utility function which determines which states are preferable and which are not on a continuous or many-valued scale. Learning agent: An agent whose performance can improve with experience.*

4. (Exercise 2.6) This exercise explores the differences between agent functions and agent programs.

   (a) Can there be more than one agent program that implements a given agent function? Give an example, or show why one is not possible. *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.

(b) Are there agent functions that cannot be implemented by any agent program? *Yes. See 1 (e)*

(c) 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.*

(d) 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^{2^n}, 2^n *internal states and a choices for each state*

(e) 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.*

5. (Exercise 3.2) Your goal is to navigate a robot out of a maze. The robot starts in the center of the maze facing north. You can turn the robot to face north, east, south, or west. You can direct the robot to move forward a certain distance, although it will stop before hitting a wall.

(a) Formulate the problem. How large is the state space?
Initial State: At((0,0)), Facing((0,1)).
Successor Function(At(x), Facing(y)):
<Turn(North), {At(x),Facing((0,1))} >
<Turn(East), {At(x),Facing((1,0))} >
<Turn(South), {At(x),Facing((0,-1))} >
<Turn(West), {At(x),Facing((-1,0))} >
<Move(k blocks), $\{At(x + y \min(k, D_{max}(x, y))),$ *Facing(y)}> where* $D_{max}(x, y)$ *is the maximum distance the robot can move in direction y from point x without hitting a wall.*
Goal State: At(x), $x \in G$, *where G is the set of locations outside the maze.*
If the maze is comprized of S blocks, then the total number of states is $4S$.

(b) In navigating a maze, the only place we need to turn is at the intersection of two or more corridors. Reformulate this problem using this observation. How large is the state space now? *The successor function remains the same for intersections, and for locations x which are straight corridors:*
Successor Function(At(x), Facing(y)):
<Move(k blocks), $\{At(x + y \min(k, D_{max}(x, y))),$ *Facing(y)}>*
Thus if the maze has I intersection blocks then the size of the state space is $4I + 2(S - I)$.

(c) From each point in the maze, we can move in any of the four directions until we reach a turning point, and this is the only action we need to do. Reformulate the problem using these actions. Do we need to keep track of the robot's orientation now? *For intersections x,*
SucessorFunction(At(x)):
<Move(North), At($x + (0,1)D_{min}(x,(0,1))$)>
<Move(East), At($x + (1,0)D_{min}(x,(1,0))$)>
<Move(South), At($x + (0,-1)D_{min}(x,(0,-1))$)>
<Move(West), At($x + (-1,0)D_{min}(x,(-1,0))$)>
where $D_{min}(x,y)$ is the minimum distance from x to an intersection in the y direction. We no longer need to keep track of the robot's orientation since the new actions now contain the turning motions within them. The total number of states is now I.

(d) In our initial description of the problem we already abstracted from the real world, restricting actions and removing details. List three such simplifications we made. *(1) The robot can move only in one of four directions. (2) The robot can sense walls perfectly. (3) After attempting to move a certain distance, the robot knows with certainty how far it has moved.*

6. (Exercise 3.5) Consider the n-queens problem using the "efficient" incremental formulation given on page 72 (page 67 RN 2nd ed.). Explain why the state space has at least $\sqrt[3]{n!}$ states and estimate the largest n for which exhaustive exploration is feasible. (*Hint*: Derive a lower bound on the branching factor by considering the maximum number of squares that a queen can attack in any column.)
We want a lower bound on the size of the state space of this formulation of the n-queens problem. In this formulation, each column contains a queen, and queens are filled in neighboring columns in locations that are not attacked by previous queens. Thus, we can lower bound the number of valid states a queen may fit in by assuming each queen from the previous columns attacks exactly 3 squares, and that none of these squares have been attacked by a previous queen (which of course may be true in a real situation, but we are only considering a lowerbound here, ie we know there are at least $n-6$ squares that the 3rd queen can take on.) Thus there will be n choices for the first queen, $n-3$ choices for the second, $n-6$ the third and so on. So if S is the size of the state space, then we have $\prod_{i=0}^{\lceil n/3 \rceil - 1} n - 3i \leq S$. Thus,

$$
\begin{aligned}
&\prod_{i=0}^{\lceil n/3 \rceil - 1} (n-3i) \\
&= \left(\prod_{i=0}^{\lceil n/3 \rceil - 1} (n-3i)^3 \right)^{1/3} \\
&\geq \left((n - 3(\lceil n/3 \rceil - 1))! \prod_{i=0}^{\lceil n/3 \rceil - 2} (n-3i)(n-3i-1)(n-3i-2) \right)^{1/3} \\
&= (n!)^{1/3} = \sqrt[3]{n!}
\end{aligned}
$$

Thus we have $\sqrt[3]{n!} \leq S$. *This becomes infeasible around* $n = 30$.