Glossary:

"Agent is anything that can perceive its *environment* through sensors and acting upon the environment through actuators. For instance, a robotic agent might have cameras and infrared range finders for sensors and various motors for actuators. While a software agent receives file contents, network packets, and human input (keyboard/mouse/…) as sensory inputs and acts on the environment by writing files, sending network packets, or displaying information, etc" [1].

"The **environment** could be everything—the entire universe! In practice it is just that part of the universe whose state we care about when designing this agent—the part that affects what the agent perceives and that is affected by the agent's actions" [1].

A **fully observable** environment is an environment, where the agent's sensors give it access to the complete state of the environment at each point in time [1].

A **stochastic** environment where the agent has a set of actions in state *s* instead of a single action, and for each action there is a probability, for instance, *A(s)* = {go up (50%), go down (35%), go straight (15%)}

A set of possible **states** that the environment can be in. We call this the state space, and it can be finite and infinite.

The **actions** available to the agent. Given a state *s*, A(*s*) returns a finite or infinite set of actions that can be executed in *s*. We call this the action space.

A **transition model**, which describes what each action does. R(*s,a*) returns the rewards from doing action *a* in state *s*. We will assume that transitions are **Markovian**: the probability of reaching *s'* from *s* depends only on *s* and not on the history of earlier states [1].

Markov Decision Problem (MDP) is a class of sequential decision problems formalized by Richard Bellman. MDP consists:

1. a set of states S.
2. a set A(s) of actions in each state.
3. a transition model *P(s'|s,a)*, the probability of ending up in state *s'* when taking action *a* in state *s*.
4. a reward function *R(s,a,s')*

You made a glossary! Very nice!!

Task 1:

Chess as MDP:

1. The size of the possible states space = 8x8x13 (6 kind of pieces per side+empty)+ additional variables for e.g. which rooks/king already did move etc. Some states can be terminal states (win/lost/remis) (some only with some probability)
2. The actions set in each state, for each piece we need to specify its role (a pawn or a knight), where it is located and if there are other pieces blocking the piece's move. An action in chess [2]:
    . selecting the piece to move
    a. then selecting among the legal moves for that piece.

So the set of all actions would include all actions that each figure can take.

3. The transition model, namely the rules of chess and the positions of the pieces.
4. Reward functions, for instance, draw 0.5, for winning 1, for losing -1.

The policy π in chess: Capture the king of the opponent player and save yours.

Task 2:

1. Agent is the lander.
2. Set of states: States consist of 8 variables, all (most) combinations of their ranges make up the state space:
    . the coordinates of the lander.
    a. the velocity of the lander
    b. its angle.
    c. its angular velocity.
    d. two booleans to represent whether each leg is in contact with the ground.
3. Set of the actions: fire left engine, fire right engine, do nothing, fire main engine.
4. Rewards:
    - Reward for moving from the top of the screen to the landing pad and coming to rest is about 100-140 points.
    - If the lander moves away from the landing pad, it loses reward.
    - If the lander crashes, it receives an additional -100 points. If it comes to rest, it receives an additional +100 points.
    - Each leg with ground contact is +10 points.
    - Firing the main engine is -0.3 points each frame. Firing the side engine is -0.03 points each frame.
1. Policy: Solved is 200 points indicating that the ship landed between the two flags without crashing.

really good and well-structured answer!

Task 3:

- Reward function r(s,a)= E[Rt+1] = Rt+1[Rt+1p(Rt+1|st,at)], we use the expected reward of performing an action rather than the actual rewards, because we want to know the average reward we get when performing an action rather than an actual reward for a specific trajectory. For instance, a self-driving car should arrive on time at a certain location, sometimes there would be a traffic jam or the car would stop at more red lights. Thus, we want to know the average reward since the environment is not constant and interacts differently with the agent. Another example; crossing the street safely with no traffic lights, the agent must look left and right to check if there is any car approaching. The agent checks right and left and assures that there are no cars, and while crossing the street an airplane falls and smashes the agent. In this specific case the agent did not arrive safe to the other side of the street, even though in another try the agent did arrive safe performing the exact same actions. Thus, we want to take the average reward, i.e, the expected reward rather than a specific reward. It is nice that you give examples

- **Are the environment dynamics generally known and can practically be used to solve a problem with RL?**
  No, the environment dynamics generally are unknown or partially known. The agent generally doesn't know the returns it receives when performing an action in a certain state and how this action in this state affects the future rewards. It cannot be a

practical solution because most of the time the agent doesn’t know the reward function or the transition model, and it is impractical to compute all the values of the state-value functions for all the states, especially when there is a large number of states with actions of continuous values.

Not only the agent doesn't know the environment dynamics, also for the programmer it is hard to predict

References:

1. Russell, S., & Norvig, P. (2020). Artificial Intelligence: A Modern Approach (4th Edition). Pearson Education (US). https://bookshelf.vitalsource.com/books/9780134671932

All in all: Very nice homework! We have nothing major to criticize. You didn't even miss the references!