COEN 266 Artificial Intelligence

Homework #1

Name: Quan Bach ID: W1561003

Problem 1:

Agent: agent is an entity that have sensors to perceive its surrounding environment, and has actuators to take actions.

Agent function: is a function that map the percept sequence to an action or a series of actions.

Agent program: is the implementation of an agent function. This implementation includes the use of hardware and software.
Performance measure: is the rating system to determine how rational an agent is.

Rational agent: is an agent that can choose an action or actions from any given percept sequence that maximize the expected performance measure.

Problem 2:

Taxi driving:

Taxi driving environment is partially observable since the taxi driver bot may see all of its surrounding plane (assuming the car has multiple cameras and infrared sensors) but it cannot observe from above.

This environment is non-deterministic because it depends not solely on the agent action. The action of the agent might even be insignificant.

This environment is sequential since the act of driving takes constant steps which depend on each other.

This environment is dynamic since it will keep changing with time.

This environment is continuous since location and time of driving vary.

This environment has multi-agents in it since there are many vehicles in traffic.

Playing soccer:

This environment is fully observable since the soccer game is enclosed in the field with certain rules and the amount of players on the field.

This environment is deterministic because its state depends on the action of the agent.

This environment is sequential if we look at within the match, the agents have to continuously play the game for 90 mins with 15 mins break in the middle.

This environment is dynamic if the game happens outdoor since this will take into account the weather. It's static if the game happens indoor.

This environment is continuous because there are infinite number of actions the agent can take as long as it follows strictly the rules of the game. The states of the game is infinite since there are no limited on the number of scores, or fouls.

This environment has multiple agents in it since it requires at least 22 players to play a soccer game.

Backgammon:

This environment is fully observable since it is within the board of the game and 2 dice.

This environment is stochastic because it involves playing with dice.

This environment is sequential since the game progress based on the actions of the agents.

This environment is static since it does not change over time.

This environment is discrete because there are limited moves the agent can make, only 2 dice involve so the numbers that show up after a toss are limited.

This environment is multi-agent since it takes 2 agents to play the game.

Problem 3:

Explanation of defined function:

Function *roomba_round()* takes a state list and an empty list as arguments. This function will perform alteration on the state list and return the state list, the action list and the number of actions taken by the robot. Its function is to change the state of the left and right room to clean (0) if they are dirty (1). It also records the action done by the robot to an action list – the empty list that taken as the second parameter.

Explanation of the test case:

The test case is a list of [1,1,0], this list indicates that the left and the right room are both dirty and the robot location is in the left room.

The function i.e. the robot will first scan if the room it's in is clean or dirty, and 'suck dust' is taken if the room is dirty. Then it will go to the other room – the right room in this case (1), and perform the same steps. Then it will check both room to see if they are both clean and terminate.

The expected steps of the robot will be: Suck dust, move right, suck dust, move left. The anticipate result matches the test outcome in this case.

Appendix: Source code

```
#Quan Bach
#COEN 266 Artificial Intelligence HW1


possible_actions = ["Room is clean. beep beep", "Suck dust."]
possible_movement =["Move right.", "Move left."]

def roomba_around(_state,_action_sequence):
	finished_cleaning = False
	number_of_actions = 0
	while (~finished_cleaning):
		roomba_location = _state[2] #get the location of the robot at the current state
		if _state[roomba_location] == 1:
			_action_sequence.append(possible_actions[_state[roomba_location]])
			number_of_actions += 1
			_action_sequence.append(possible_movement[roomba_location])
			number_of_actions += 1
			_state[roomba_location] = 0
			if roomba_location == 0:
				_state[2] = 1
			else:
				_state[2] = 0
		else:
			_action_sequence.append(possible_movement[roomba_location])
			number_of_actions += 1
			if roomba_location == 0:
				_state[2] = 1
			else:
				_state[2] = 0


		if (number_of_actions > 0 and _state[0] == 0 and _state[1] == 0):
			finished_cleaning = True
			return _state, _action_sequence, number_of_actions
```

```
test_case = [1,1,0]
print("Test case: ", test_case)
roomba_action = []
test = roomba_around(test_case,roomba_action)
print(test)
```