**Question 1: This question is based on the following scenario:**

**UQ Soda is a very small soft drink seller at UQ. It sells two types of soft drinks: Coke and Sprite. It buys its stock each morning, but can only buy at most 2 cans of soda per day and can only store and sell 3 cans of soda per day. You can assume that:**

* **Coke and Sprite are the only two types of sodas that the customer may want to buy**
* **The customer will not change its preference and will leave UQSoda without buying anything if his/her choice is not available**
* **The sodas will always be in a good condition**
* **The customers preference depends on the amount of each type of sodas available at the beginning of the day (right after UQSoda buys its stock)**

**Despite of its size and lack of information about the behaviour of its consumer, UQSoda wants to stock the sodas such that it can minimize the number of customers who could not get their choice of sodas**

1. **The stocking problem of UQ Soda in Question 3 Tutorial 8 is an MDP problem. However, the stocking problem in the above scenario is a reinforcement learning problem, even though the problem is similar to the scenario in Question 3 of Tutorial 8. Please explain the differences and similarity between MDP and Reinforcement Learning. In your explanation, please provide a comparison example based on the UQSoda stocking problems as described above and as described in Tutorial 8.**
2. **Suppose the owner of UQSoda wants to solve the reinforcement learning stocking problem they face using model based Bayesian Reinforcement Learning problem (yes, he knows there’s such an approach). The owner knows that this means he has to frame the problem as a POMDP problem, but does not really know what a POMDP is. To help him, your task is to define the POMDP problem that represents the UQSoda stocking problem (as described in this assignment)**

*Answer:*

**similarities and differences between MDP and Reinforcement Learning**

Firstly the similarities. Both have a state and action space, they still need transition and reward functions and they also still need to generate an optimal policy. In both Tutorial 8 and the question above, we are given enough information to create the State and Action space.

*The different types of soda we sell, the total number of cans we can purchase and the total number of cans we can store and sell help us generate the State and Action space. These are both available in the question above and the stocking problem given in Tutorial 8.*

Now for the differences. In MDP, we have enough information to create the Transition and Reward Function. In Tutorial 8, we are given the probabilities for user consumption for each type of soda. This helps us generate the Transition and Reward Function. Because of this, we can use MDP to solve the problem. In Reinforcement Learning, we do not have information to create the Transition and Reward Function. In this question, we are not given the probabilities for user consumption of each type of soda. This meant that we do not know the user’s behaviour, so we cannot create these Transition and Reward Functions. Because of this, we need Reinforcement Learning.

**POMDP Problem Definition**

* 1. State Space (S)
     1. State Space is a tuple of 2 elements representing the amount of cans for a particular type of soda currently stocked for a day before purchasing for additional stock
     2. Formally: S = <S0, S1> where S0 is the number of Coke cans stored and S1 is the number of Sprite cans stored and C0 + C1 <= 3.
  2. Action Space (A)
     1. Action Space is a tuple of 2 elements representing the amount of cans for a particular type of soda that was purchased for stock in a day.
     2. Formally: A = <P0, P1> where P0 is the number of Coke cans purchased and P1 is the number of Sprite cans purchased and P0 + P1 <= 2
  3. Observation Space (Ω)
     1. The observation space is the amount of cans consumed for each soda in a day
     2. Formally: O = <C0, C1> where C0 is the number of Coke cans consumed by users and C1 is the number of Sprite cans consumed by users and C0 + C1 <= 3. This is assuming that the number of cans users can consume in a day is equal to the number of cans we can purchase
  4. Transition Function (T)
     1. Assuming that the MDP model is as described by POMDP state s S
     2. Given the previous state st, the action taken at and the new state st + 1, where t = current time slice, st, st + 1 S and at A the transition function will return the probability of ending up at st + 1 after doing at at state st.
     3. Informally, it will be the probability of ending up with a particular number of cans for each soda type after purchasing soda.
  5. Observation Function (Z)
     1. Given the new state st + t, the action taken at and the observation ot, where t = current time slice, st + 1 S and ot O, the observation function will give the probability of ending up at st + 1 in relation to the amount of soda consumed by users
  6. Reward Function (R)
     1. Assuming that the MDP model is as described by POMDP state s S
     2. Given the new state s and the action taken a, where s S and a A, the reward function will give us the reward for making it to s after doing a.
     3. Informally, we will be rewarded depending on how much sodas we have left at the end of the day. For this example, we can assume that if we have 0 cans of soda that we possibly failed with providing for the user.
     4. Therefore, our rewards will be as follows:
        1. Neither of our sodas are out of stock => +10
        2. Out of stock for either Coke or Sprite => 0
        3. Out of stock for both => -5

**Question 2:**

**Let’s consider Q-learning and SARSA with greedy algorithm for action selection. Please explain the effect of different values on the performance of the learning methods. In particular, please explain the effect when = 0, = 1 and as increases from 0 to 1**

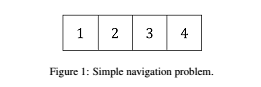
*Answer:*

1. The epsilon-greedy algorithm involves assigning weight to each action. Changing the value of determines the probability of selecting an action to generate the Q value for each episode. Setting = 0 means that the action with the highest weight will be selected with probability 1 while all other actions are selected with probability 0. Setting = 1 means that the action with the highest weight will be selected with probability 0 while all other actions are selected with probability 1. Hence, setting at extremes determines the priority meant by the weight. (If high weight, best action if = 0, worst action if = 1.
2. This means that as increases from 0 to 1, we start placing priority on the other possible actions. Starting at = 0, we have only selected the best action, but as we increase , each action gets a higher and higher chance of being evaluated.
3. Knowing this, we can check the effect of the values of on Q-Learning and SARSA
   1. Q-Learning
      1. = 0
         1. Only observe the reward and the next state after taking the most favoured action.
         2. Base the Q-value solely on the best action. Hence, we are choosing to do the most favoured action for every single state. When calculating the new Q-value, we pick the action with the largest Q-value (a \* [r + discount factor \* maxa’ \* Q(s’, a’) – Q(s, a)]), but since we only pick the most favoured action when changing the Q-value, other actions may never be selected, depending on their initialized value
      2. = 1
         1. Observe every action except the most favoured action
         2. Base the Q-value on a range of actions. When calculating the new Q-value, we have a range of actions to choose from, in contrast to when = 0
      3. increases from 0 to 1
         1. If we start at = 0, the first few episodes (or the first few steps of the first episode) will have a Q-value dependent on the most favoured action
         2. But as we increase , we start taking into account other actions. As a result, the other episodes have a wider variety of actions to take.
         3. Once = 1, the most favoured action is no longer taken into account in calculating the Q-value.
   2. SARSA
      1. = 0
         1. Similar to Q-Learning
         2. Difference is that we are setting the entire policy to be the most favoured action, where in Q-Learning we are simply setting the Q-value. This is regardless of what state we are in
      2. = 1
         1. Similar to Q-Learning again, except now the policy will be any one of the less favoured actions. The most favoured action will not be a part of the policy at all
      3. increases from 0 to 1
         1. Once again, similar to Q-Learning, but now, every single action now has a possibility of being a part of the policy.
         2. We start at = 0. Initially, the policy will state that we use the most favoured action.
         3. As we increase , the policy will tell us to take the less favoured actions.
         4. Once = 1, the most favoured actions is no longer taken into account in calculating the Q-value and we will only select the less-favoured actions

**Question 3:**

**This question is based on the following simple navigation scenario.**

**A robot is navigating in an environment that has been discretised into 4 grid cells, as shown in Figure 1. We know the robot’s action space is {left, right} and that effect of performing an action is non-deterministic. However, we do not have a stochastic model of this non-deterministic behaviour of the robot. Worse, we do not know the cost of each action that the robot takes nor the reward for reaching the goal. Despite this lack of information, we want to find a policy for the robot to move from cell-1 to cell-4 with minimum cost.**



**Suppose we want to solve this problem using model-free reinforcement learning approaches. We set the initial values of all states to be zero, generated an initial policy and used this policy to generate the following two episodes.**

**Episode-1: cell1-right- -1, cell1-right- -1, cell2-right- -1, cell2-right - -1, cell3-right-9, cell4**

**Episode-2: cell1-right- -1, cell2-right- -1, cell2-right- -1, cell3-right-9, cell4**

**Now we want to evaluate the policy . Please write down the value functions of the first two iterations of the following model-free reinforcement learning methods:**

1. **TD(0)**
2. **TD(2)**
3. **Monte Carlo**

*Answers:*

1. Before starting, we need to set/know the following values
   1. Discount Factor ()
      1. No discount factor given so arbitrarily set the discount factor to be 1 to simplify calculations
   2. States
      1. We only have 4 states. These are Cell1, Cell2, Cell3 or Cell4
   3. value
      1. The value is not given, so let’s arbitrarily set it to be 1 to simplify calculations
   4. Reward
      1. Each reward is given after each action
   5. Initial values for all states
      1. All states start with a value of 0
2. TD(0) Answer
   1. Episode 1
      1. 1st Step (Cell1 – right - -1 -> Cell1)
         1. V(Cell1) = 0 + 1(-1 + 0 – 0) = -1
      2. 2nd Step (Cell1 – right - -1 -> Cell2)
         1. V(Cell1) = -1 + 1(-1 + 0 - -1) = -1
      3. 3rd Step (Cell2 – right - -1 -> Cell2)
         1. V(Cell2) = 0 + 1(-1 + 0 – 0) = -1
      4. 4th Step (Cell2 – right - -1 -> Cell3)
         1. V(Cell2) = -1 + 1(-1 + 0 - -1) = -1
      5. 5th Step (Cell3 – right – 9 -> Cell4)
         1. V(Cell3) = 0 + 1(9 + 0 – 0) = 9
      6. Values For States
         1. V(Cell1) = -1
         2. V(Cell2) = -1
         3. V(Cell3) = 9
         4. V(Cell4) = 0
   2. Episode 2
      1. 1st Step (Cell1 – right - -1 -> Cell2)
         1. V(Cell1) = -1 + 1(-1 + -1 - -1) = -1 + 1(-1) = -2
      2. 2nd Step (Cell2 – right - -1 -> Cell2)
         1. V(Cell2) = -1 + 1(-1 + -1 - -1) = -2
      3. 3rd Step (Cell2 – right - -1 -> Cell3)
         1. V(Cell2) = -2 + 1(-1 + 9 - -2) = 8
      4. 4th Step (Cell3 – right – 9 -> Cell4)
         1. V(Cell3) = 9 + 1(9 + 0 – 9) = 9
      5. Values For States
         1. V(Cell1) = -2
         2. V(Cell2) = 8
         3. V(Cell3) = 9
         4. V(Cell4) = 0
   3. Final Values
      1. V(Cell1) = -2
      2. V(Cell2) = 8
      3. V(Cell3) = 9
      4. V(Cell4) = 0
3. TD(2) Answer
   1. Episode 1
      1. 1st Step (Cell1 – right - -1 -> Cell1)
         1. V(Cell1) = 0 + 1(-1 + 1(-1) + 12(-1) + 13(0) - 0) = -3
      2. 2nd Step (Cell1 – right - -1 -> Cell2)
         1. V(Cell1) = -3 + 1(-1 + -1 + -1 + 0 - -3) = -3
      3. 3rd Step (Cell2 – right - -1 -> Cell2)
         1. V(Cell2) = 0 + 1(-1 + -1 + 9 + 0 – 0) = 7
      4. 4th Step (Cell2 – right - -1 -> Cell3)
         1. V(Cell2) = 7 + 1(-1 + 9 + 0 + 0 – 7) = 8
      5. 5th Step (Cell3 – right – 9 -> Cell4)
         1. V(Cell3) = 0 + 1(9 + 0 + 0 + 0 – 0) = 9
      6. Values For States
         1. V(Cell1) = -3
         2. V(Cell2) = 8
         3. V(Cell3) = 9
         4. V(Cell4) = 0
   2. Episode 2
      1. 1st Step (Cell1 – right - -1 -> Cell2)
         1. V(Cell1) = -3 + 1(-1 + -1 + -1 + 8 - -3) = 5
      2. 2nd Step (Cell2 – right - -1 -> Cell2)
         1. V(Cell2) = 8 + 1(-1 + -1 + 9 + 8 – 8) = 15
      3. 3rd Step (Cell2 – right - -1 -> Cell3)
         1. V(Cell2) = 15 + 1(-1 + 9 + 0 + 9 – 15) = 17
      4. 4th Step (Cell3 – right – 9 -> Cell4)
         1. V(Cell3) = 9 + 1(9 + 0 + 0 + 0 – 9) = 9
      5. Values For States
         1. V(Cell1) = 5
         2. V(Cell2) = 17
         3. V(Cell3) = 9
         4. V(Cell4) = 0
   3. Final Values
      1. V(Cell1) = 5
      2. V(Cell2) = 17
      3. V(Cell3) = 9
      4. V(Cell4) = 0
4. Monte Carlo Answer
   1. Episode 1
      1. 1st Step (Cell1 – right - -1 -> Cell1)
         1. V(Cell1) = 0 + 1(-1 + -1 + -1 + -1 + 9 – 0) = 5
      2. 2nd Step (Cell1 – right - -1 -> Cell2)
         1. V(Cell1) = 5 + 1(-1 + -1 + -1 + 9 - 5) = 6
      3. 3rd Step (Cell2 – right - -1 -> Cell2)
         1. V(Cell2) = 0 + 1(-1 + -1 + 9 – 0) = 7
      4. 4th Step (Cell2 – right - -1 -> Cell3)
         1. V(Cell2) = 7 + 1(-1 + 9 – 7) = 8
      5. 5th Step (Cell3 – right – 9 -> Cell4)
         1. V(Cell3) = 0 + 1(9 – 0) = 9
      6. Values For States
         1. V(Cell1) = 6
         2. V(Cell2) = 8
         3. V(Cell3) = 9
         4. V(Cell4) = 0
   2. Episode 2
      1. 1st Step (Cell1 – right - -1 -> Cell2)
         1. V(Cell1) = 6 + 1(-1 + -1 + -1 + 9 – 6) = 6
      2. 2nd Step (Cell2 – right - -1 -> Cell2)
         1. V(Cell2) = 8 + 1(-1 + -1 + 9 – 8) = 7
      3. 3rd Step (Cell2 – right - -1 -> Cell3)
         1. V(Cell2) = 7 + 1( -1 + 9 – 7) = 8
      4. 4th Step (Cell3 – right – 9 -> Cell4)
         1. V(Cell3) = 9 + 1(9 - 9) = 9
      5. Values For States
         1. V(Cell1) = 6
         2. V(Cell2) = 8
         3. V(Cell3) = 9
         4. V(Cell4) = 0
   3. Final Values
      1. V(Cell1) = 6
      2. V(Cell2) = 8
      3. V(Cell3) = 9
      4. V(Cell4) = 0