PS12

Graded

Student

Chetan Hiremath

Total Points

72 / 100 pts

Question 1

12.1 — **10** / 10 pts

- ✔ **+ 10 pts** Correct

Question 2

12.2 — **10** / 10 pts

- ✔ **+ 10 pts** Correct

Question 3

12.3 — **9** / 10 pts

3.1 (a) — **5** / 5 pts

- ✔ **+ 5 pts** Correct

3.2 (b) — **4** / 5 pts

- ✔ **+ 5 pts** Correct
- ✔ **− 1 pt** Incorrect value or incorrect contour line

Question 4

12.4 — **16** / 20 pts

4.1 (a) — **10** / 10 pts

- ✔ **+ 10 pts** Correct

4.2 (b) — **6** / 10 pts

- ✔ **+ 10 pts** Correct
- ✔ **− 2 pts** Too many updates per episode
- ✔ **− 2 pts** incorrect/missing final Q values

Question 5

12.5 — **8** / 10 pts

5.1 (a) — **5** / 5 pts

- ✔ **+ 5 pts** Correct

5.2 (b) — **3** / 5 pts

- ✔ **+ 5 pts** Correct
- ✔ **− 2 pts** Incorrect calculation for E

Question 6

12.6 — **3** / 10 pts

6.1 (a) — **2** / 5 pts

- ✔ **+ 5 pts** Correct
- ✔ **− 1 pt** Incorrect equation setup
- ✔ **− 2 pts** incorrect

6.2 (b) — **1** / 5 pts

- ✔ **+ 5 pts** Correct
- ✔ **− 2 pts** Missing conclusion
- ✔ **− 2 pts** Incorrect / Faulty reasoning / incomplete

Question 7

12.7 — **16** / 30 pts

- ✔ **+ 30 pts** Correct
- ✔ **− 7 pts** Incorrect/Missing output A
- ✔ **− 7 pts** Incorrect/Missing output B

Question assigned to the following page: 1

1. The problem is that the communication is not effective. The goal of the agent in Reinforcement Learning is to maximize the expected total reward and escape from the maze. But the agent is not making any significant progress because the agent is not trained to leave the maze. It doesn't know any reward values even though it has a reward value of +1 and a reward value of 0. But it doesn't have other reward values for detecting the wrong states and finding the optimal path. One way to train the agent efficiently is to add -1 as another reward value for a state in the maze. So, the agent can ignore states with negative reward values and pick states with non-negative reward values to find the goal state. Then, the agent can successfully escape from the maze.

Question assigned to the following page: 5.1

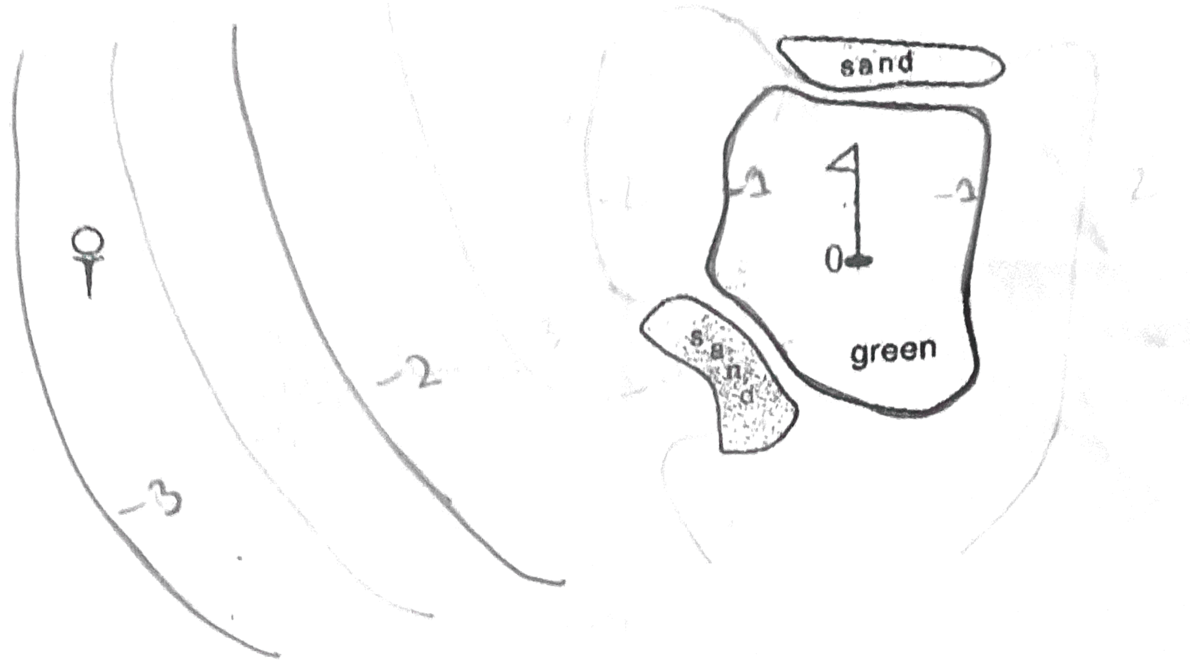
5a.

P1 \ P2	R	P	S
R	0, 0	-1, +1	+1, -1
P	+1, -1	0, 0	-1, +1
S	-1, +1	+1, -1	0, 0

Question assigned to the following page: 3.1

3a.

$V^*(s)$



Question assigned to the following page: 6.1

6a.

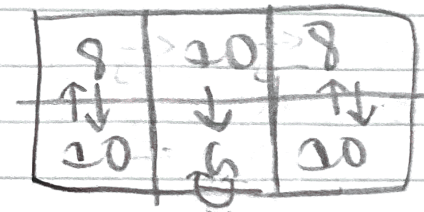
P1 P2	H	T
H	\$1 million	-\$1 cent
T	-\$1 cent	\$1 cent

Minimax = -\$1 cent.
Maximin = \$1 million.

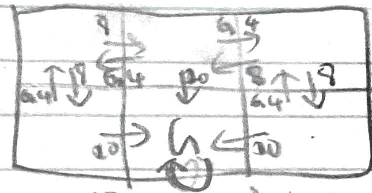
Player 1 should choose H with probability of $\frac{999999}{2}$.
Player 2 should choose H with probability of $\frac{999999}{2}$.
The mixed strategy produces a saddle solution since minimax $\neq$ maximin.

Question assigned to the following page: 4.1

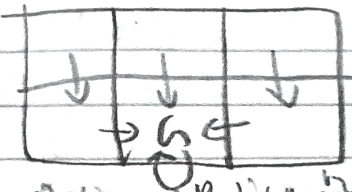
4a.



V^*



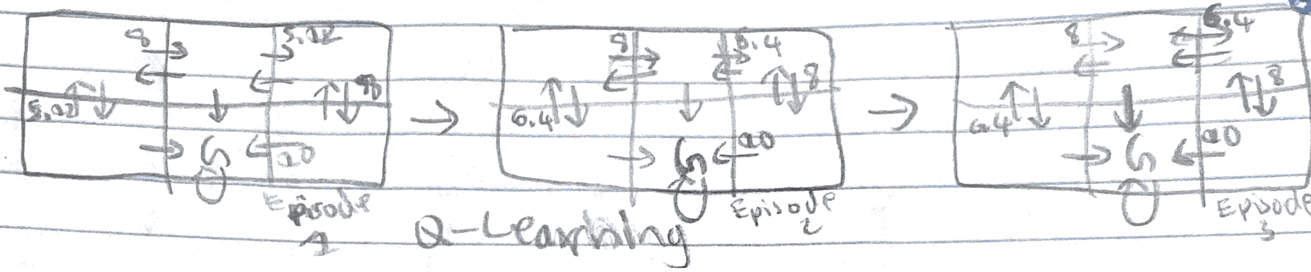
Q(s, a)



Optimal Policy with
$\gamma = 0.8$

Question assigned to the following page: 4.2

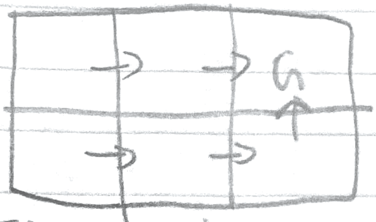
b.



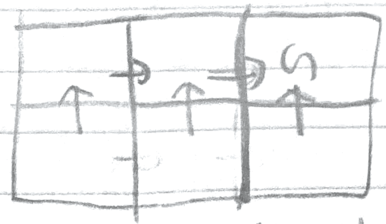
Q-Learning

Question assigned to the following page: 2

2.



First Optimal Policy



Second Optimal Policy

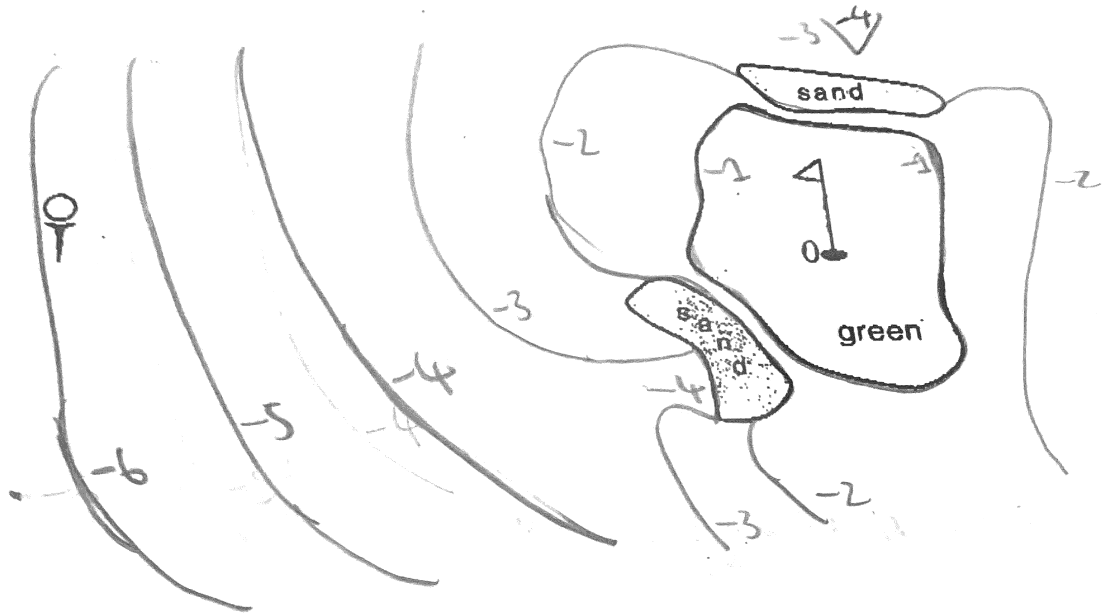
1	2	6
3	4	5

There are 4 optimal policies in this MDP.

Question assigned to the following page: 3.2

b.

$Q^*(s, \text{putt})$



b. Player 1 = $\frac{1}{3}(0-1+1)+\frac{1}{3}(1+0-1)+\frac{1}{3}(-1+1+0)$
$= 0$. Player 1 Payoff = Player 2 Payoff.
Minimax for column = max(-1, -1, -1) = -1.
Maximin for row = min(1, 1, 1) = 1.
So, the mixed strategy produces a saddle solution since minimax ≠ maximin.

Question assigned to the following page: 6.2

b. $b=c$

$a=2b-c+d$

Players choose H by these conditions.

$a=d$.

$b=-c$.

Players choose T by these conditions.

Question assigned to the following page: 7

7a.

3	0.81	0.90	1	+1
2	0.73		0.90	−1
1	0.66	0.73	0.81	0.73
	1	2	3	4

Optimal Policy Values

b.

3	-0.62	-0.46	0.47	+1
2	-0.67		-0.52	−1
1	-0.73	-0.76	-0.64	-0.73
	1	2	3	4

1 2 3 4

Random Policy Values

Question assigned to the following page: 7

7a. I have used and run the program of policy evaluation from the Python code of Mdp.ipynb and AIMA Python File: mdp.py on the optimal policy when it uses R = -0.04 and gamma = 1. I have tried 1000 trails since these trails will allow me to find the optimal policy. Then, I have compared the program's answers and R&N Textbook's Figure 22.1(b)'s answers. They are approximately same and accurate, and I have recorded these answers on the top grid of the linked sheet.

Question assigned to the following page: 7

b. I have modified the program of the Python code of Mdp.ipynb and AIMA Python File: mdp.py to learn the random policy when it uses R = -0.04 and gamma = 1. One non-terminal state chooses actions like Up, Down, Left, and Right. These actions' probabilities are equal, and I have used 1000 trails since I can find the random policy easily. Here are the modified parts that are used in the Python code of Mdp.ipynb and AIMA Python File: mdp.py since these modified parts have allowed me to find the approximate and accurate results. Then, I have recorded these answers on the bottom grid of the linked sheet since the random policy is found successfully in 1000 trails.

```
def pDUE(mdp, trails=1000, alpha=0.1):
    util = {state: 0 for state in mdp.states}
    counts = {state: 0 for state in mdp.states}
    for _ in range(trails):
        state = mdp.init
        while state not in mdp.terminals:
            action = random_policy(mdp, state)
            next_state, reward = random.choice(mdp.T(state, action))
            counts[state] += 1
            util[state] += (reward + mdp.gamma * util[next_state] -
util[state]) / counts[state]
            state = next_state
    return util

grid_mdp = GridMDP([[-0.04, -0.04, -0.04, 1], [-0.04, None, -0.04, -1], [-
0.04, -0.04, -0.04, -0.04]], terminals=[(3, 2), (3, 1)], gamma=1)

estimated_utilities_random = pDUE(grid_mdp)
utility_grid = grid_mdp.to_grid(estimated_utilities_random)
```