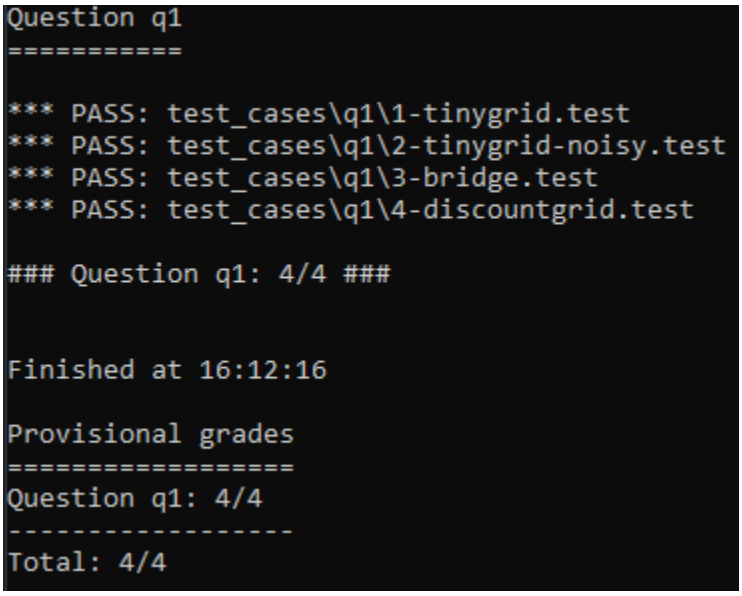
Reinforcement Learning

Tyler Gillette

1. 

Reinforcement Learning

Tyler Gillette

```
def computeQValueFromValues(self, state, action):
    """
      Compute the Q-value of action in state from the
      value function stored in self.values.
    """
    "*** YOUR CODE HERE ***"
    "get the rewards from the given state"
    def getReward(s, a, s_):
        "this will return the reward + the y discount * the prev value"
        return self.mdp.getReward(s, a, s_) + self.discount*self.getValue(s_)

    "what other states can i get to and whats the probability of gettign to them"
    statesAndProbs = self.mdp.getTransitionStatesAndProbs(state, action)
    return sum(p * getReward(state, action, s) for s, p in statesAndProbs)
    "*** YOUR CODE HERE ***"
```

```
def computeActionFromValues(self, state):
    """
      The policy is the best action in the given state
      according to the values currently stored in self.values.
      You may break ties any way you see fit.  Note that if
      there are no legal actions, which is the case at the
      terminal state, you should return None.
    """
    "*** YOUR CODE HERE ***"
    "get the q values possible."
    def theKey(a): return self.computeQValueFromValues(state, a)
    "then return the max given thos values"
    return max(self.mdp.getPossibleActions(state), key=theKey, default=None)
    "*** YOUR CODE HERE ***"
```

2.

```
Question q2
===========

*** PASS: test_cases\q2\1-bridge-grid.test

### Question q2: 1/1 ###


Finished at 16:15:39

Provisional grades
==================
Question q2: 1/1
------------------
Total: 1/1
```

Reinforcement Learning

Tyler Gillette

```python
def question2():
    answerDiscount = 0.9
    answerNoise = 0.2
    return answerDiscount, answerNoise
```

3.
```
Question q3
===========

*** PASS: test_cases\q3\1-question-3.1.test
*** PASS: test_cases\q3\2-question-3.2.test
*** PASS: test_cases\q3\3-question-3.3.test
*** PASS: test_cases\q3\4-question-3.4.test
*** PASS: test_cases\q3\5-question-3.5.test

### Question q3: 5/5 ###


Finished at 16:17:18

Provisional grades
==================
Question q3: 5/5
------------------
Total: 5/5
```

Reinforcement Learning

Tyler Gillette

```python
def question3a():
    answerDiscount = .1
    answerNoise = 0
    answerLivingReward = 0
    return answerDiscount, answerNoise, answerLivingReward
    # If not possible, return 'NOT POSSIBLE'


def question3b():
    answerDiscount = .5
    answerNoise = .2
    answerLivingReward = -2
    return answerDiscount, answerNoise, answerLivingReward
    # If not possible, return 'NOT POSSIBLE'


def question3c():
    answerDiscount = .9
    answerNoise = 0
    answerLivingReward = 0
    return answerDiscount, answerNoise, answerLivingReward
    # If not possible, return 'NOT POSSIBLE'


def question3d():
    answerDiscount = .9
    answerNoise = .2
    answerLivingReward = 0
    return answerDiscount, answerNoise, answerLivingReward
    # If not possible, return 'NOT POSSIBLE'


def question3e():
    answerDiscount = 0
    answerNoise = 0
    answerLivingReward = 10
    return answerDiscount, answerNoise, answerLivingReward
    # If not possible, return 'NOT POSSIBLE'
```

Reinforcement Learning

Tyler Gillette

4.
```
Question q6
===========

*** PASS: test_cases\q6\1-tinygrid.test
*** PASS: test_cases\q6\2-tinygrid-noisy.test
*** PASS: test_cases\q6\3-bridge.test
*** PASS: test_cases\q6\4-discountgrid.test

### Question q6: 4/4 ###


Finished at 16:18:09

Provisional grades
==================
Question q6: 4/4
------------------
Total: 4/4
```

```
def update(self, state, action, nextState, reward):
    """
      The parent class calls this to observe a
      state = action => nextState and reward transition.
      You should do your Q-Value update here
      NOTE: You should never call this function,
      it will be called on your behalf
    """

    "*** YOUR CODE HERE ***"
    "make sure the state is in the qvalues list, otherwise set it to 0"
    if state not in self.qValues:
        "state wasnt in the list, so set it to 0 cant get to it."
        self.qValues[state] = {action: 0.0}
        "set the qvalue based off the a * qval"
    self.qValues[state][action] = (1 - self.alpha) * self.getQValue(
        state, action) + self.alpha * (reward + self.discount * self.getValue(nextState))
    "*** YOUR CODE HERE ***"
```

```
def computeValueFromQValues(self, state):
    """
      Returns max_action Q(state,action)
      where the max is over legal actions.  Note that if
      there are no legal actions, which is the case at the
      terminal state, you should return a value of 0.0.
    """

    "*** YOUR CODE HERE ***"
    def q(action): return self.getQValue(state, action)
    # max(T(s,a,s')[R(s,a,s') + rVk(s')])
    return max(map(q, self.getLegalActions(state)), default=0.0)
    "*** YOUR CODE HERE ***"
```

```
def getQValue(self, state, action):
    """
      Returns Q(state,action)
      Should return 0.0 if we have never seen a state
      or the Q node value otherwise
    """

    "*** YOUR CODE HERE ***"
    "check if the states are in qvals and action in qvals"
    " question 6"
    if state in self.qValues and action in self.qValues[state]:
        return self.qValues[state][action]
    return 0.0
    "*** YOUR CODE HERE ***"
```

```
def computeActionFromQValues(self, state):
    """
      Compute the best action to take in a state.  Note that if there
      are no legal actions, which is the case at the terminal state,
      you should return None.
    """

    "*** YOUR CODE HERE ***"
    "get the qvalue given the state and action"
    def q(action): return self.getQValue(state, action)
    "retunr the max of the possible actions"
    return max(self.getLegalActions(state), key=q, default=None)
    "*** YOUR CODE HERE ***"
```

5.
```
Question q7
===========

*** PASS: test_cases\q7\1-tinygrid.test
*** PASS: test_cases\q7\2-tinygrid-noisy.test
*** PASS: test_cases\q7\3-bridge.test
*** PASS: test_cases\q7\4-discountgrid.test

### Question q7: 2/2 ###


Finished at 16:18:29

Provisional grades
==================
Question q7: 2/2
------------------
Total: 2/2
```

Reinforcement Learning

Tyler Gillette

```python
def getAction(self, state):
    """
      Compute the action to take in the current state.  With
      probability self.epsilon, we should take a random action and
      take the best policy action otherwise.  Note that if there are
      no legal actions, which is the case at the terminal state, you
      should choose None as the action.
      HINT: You might want to use util.flipCoin(prob)
      HINT: To pick randomly from a list, use random.choice(list)
    """

    "*** YOUR CODE HERE ***"
    "use the flipcoin prob, make sure its in range"
    if util.flipCoin(1 - self.epsilon):
        "return the qvalue"
        return self.computeActionFromQValues(state)

    "return the randomly selected path"
    return random.choice(self.getLegalActions(state))
    "*** YOUR CODE HERE ***"
```

6.
```
Question q8
===========

*** PASS: test_cases\q8\grade-agent.test

### Question q8: 1/1 ###


Finished at 16:18:45

Provisional grades
==================
Question q8: 1/1
------------------
Total: 1/1
```

```python
def question8():
    answerEpsilon = 'NOT POSSIBLE'
    answerLearningRate = 'NOT POSSIBLE'
    # return answerEpsilon, answerLearningRate
    # If not possible, return 'NOT POSSIBLE'
    return 'NOT POSSIBLE'
```

Reinforcement Learning

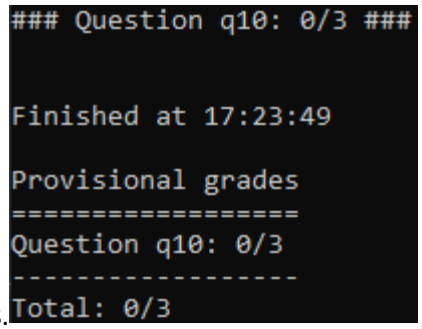
Tyler Gillette

```
def question8():
    answerEpsilon = None
    answerLearningRate = None
    # return answerEpsilon, answerLearningRate
    # If not possible, return 'NOT POSSIBLE'
    return 'NOT POSSIBLE'
```

```
def getQValue(self, state, action):
    """
      Returns Q(state,action)
      Should return 0.0 if we have never seen a state
      or the Q node value otherwise
    """

    "*** YOUR CODE HERE ***"
    "check if the states are in qvals and action in qvals"
    " question 6"
    if state in self.qValues and action in self.qValues[state]:
        return self.qValues[state][action]
    return 0.0
    "*** YOUR CODE HERE ***"
```

7.
```
### Question q9: 1/1 ###


Finished at 16:19:17

Provisional grades
==================
Question q9: 1/1
------------------
Total: 1/1
```

```
-----------------------------------------
Pacman emerges victorious! Score: 495
Pacman emerges victorious! Score: 503
Pacman emerges victorious! Score: 503
Pacman emerges victorious! Score: 495
Pacman emerges victorious! Score: 502
Pacman emerges victorious! Score: 502
Pacman emerges victorious! Score: 495
Pacman emerges victorious! Score: 495
Pacman emerges victorious! Score: 503
Pacman emerges victorious! Score: 499
Average Score: 499.2
Scores:        495.0, 503.0, 503.0, 495.0, 502.0, 502.0, 495.0, 495.0, 503.0, 499.0
Win Rate:      10/10 (1.00)
Record:        Win, Win, Win, Win, Win, Win, Win, Win, Win, Win
```

8. 
Question q10: 0/3
Finished at 17:23:49
Provisional grades
==========
Question q10: 0/3

Total: 0/3