**COMPSCI 383 – Fall 2022**

Homework 5 Coding

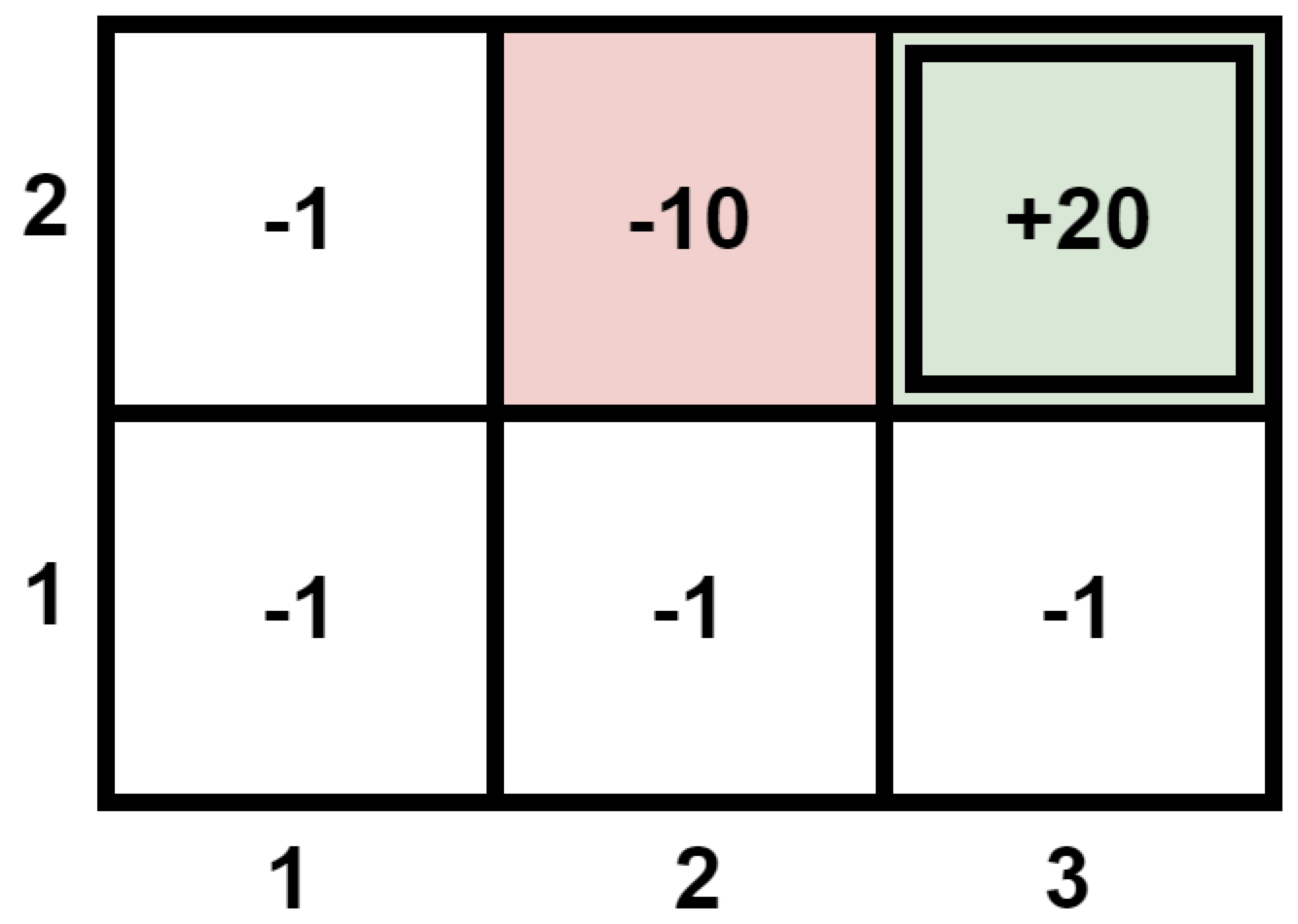
**Due Tuesday, November 29th at 11:59pm ET**

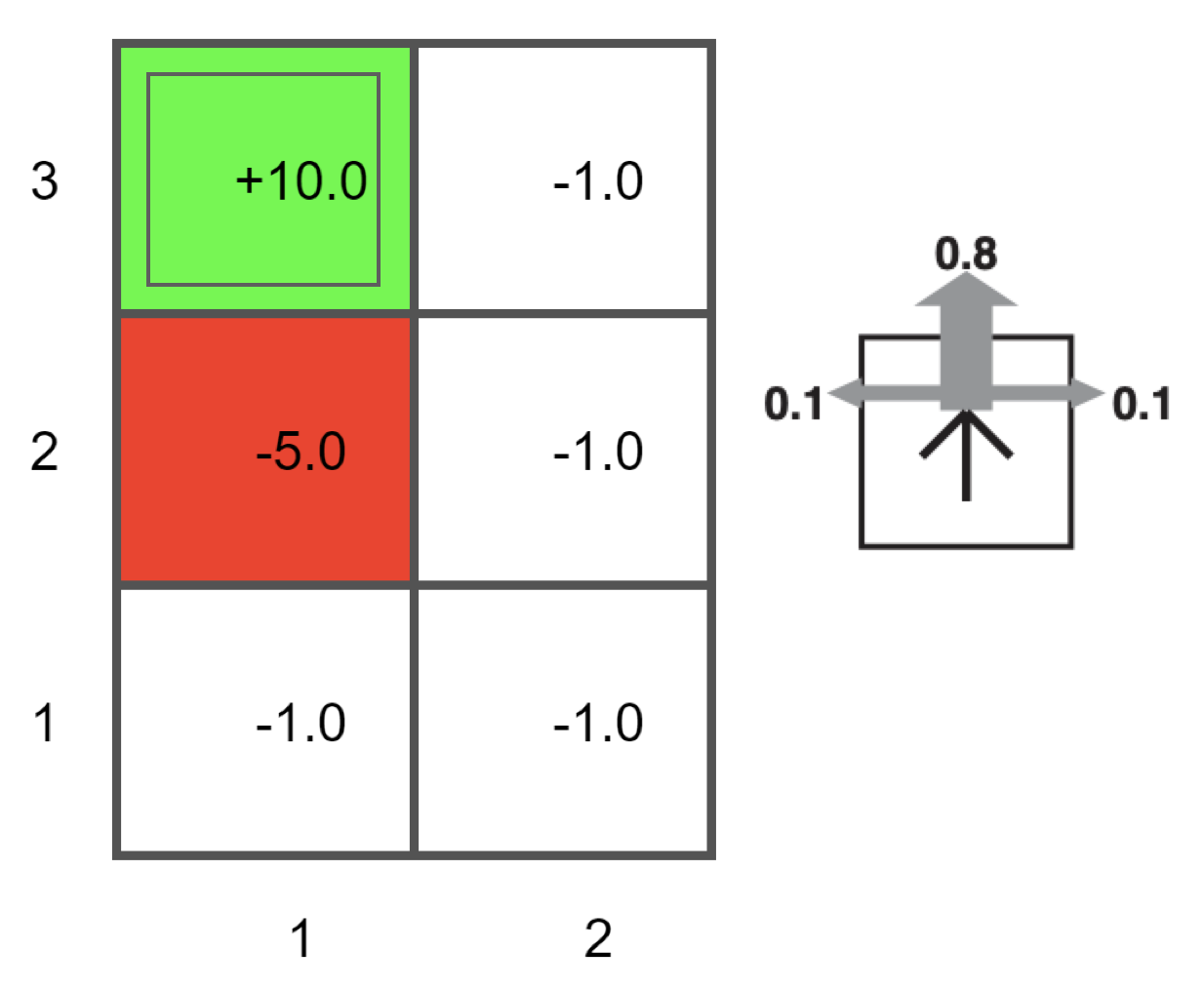
*You are encouraged to discuss the assignment in general with your classmates, and may optionally collaborate with one other student. If you choose to do so, you must indicate with whom you worked. Multiple teams (or non-partnered students) submitting the same code will be considered plagiarism.*

*Code must be written in a reasonably current version of Python (>3.8), and be executable from a Unix command line. You are free to use Python’s standard modules for data structures and utilities, as well as the pandas, scipy, and numpy modules.*

## Value Iteration and Policies for Gridworld

For this short coding assignment, you will implement Value Iteration for grid world MDPs. The goal of the assignment is to build off of your understanding of the algorithm from the Homework 5 Primer, translate it into code, and see the effects of different parameters on policies.



Recall the gridworld MDP shown on the right from the Primer. The single terminal state (3, 2) has a reward of +20, the non-terminal (2, 2) has reward -10, and all other states have a reward of -1.   
  
The agent makes its intended move (up, down, left, or right) with a probability 0.8, and moves in a perpendicular direction with probability 0.1 for each side (e.g., if intending to go right, the agent can move up or down with a probability of 0.1 each). If the agent runs into a wall, it stays in the same place.   
  
The only code you need to work on is found in mdp.py and answers.py. To make your life easier, we have supplied you with a simple MDP framework for you to work with. **Before starting your implementation, please read all the comments and docstrings in the code.**

## 1. Implementing Value Iteration (30 points)

Implement the value\_iteration()function in mdp.py. You should utilize the methods provided, but are free to add additional methods and functions as you see fit. Before coding up your solution, you should complete Question 2 in the Homework 5 Primer in order to understand the algorithm. You can use these answers to verify your code’s correctness and vice versa. Next, run your value iteration code by running the code in answers.py (imports mdp) and make sure that your implementation of value iteration in mdp.py runs without error for the given example MDP. Note that you haven’t filled in the code for the q1\_and\_2(), q3(), q4(), and q5() functions yet, so your output may say the values for them are still None.

Next, replace the None values for the variables gridworld, EPSILON, and discount\_factor in the q1\_and\_2()function in answers.py with a discount factor of = 0.8 and convergence threshold of = 0.01 and run answers.py. Fill in the utilities corresponding to different iterations of the algorithm in the table on the left below. The final row should contain values your algorithm produced after convergence. On the right, draw arrows showing the policy derived for the non-terminal states. The output from ascii\_grid\_utils() and ascii\_grid\_policy() in mdp.py may be helpful for completing the tables (gen\_results() in answers.py already calls them).

**Utilities Policy**

| s | **(1, 1)** | **(2, 1)** | **(3, 1)** | **(1, 2)** | **(2, 2)** | **2** |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **U0(s)** | -1 | -1 | -1 | -1 | -10 |
| **U2(s)** |  |  |  |  |  | **1** |  |  |  |
| **U5(s)** |  |  |  |  |  |
| **U\*(s)** |  |  |  |  |  |  | **1** | **2** | **3** |

## 2. Convergence (5 points)

How many iterations did it take for the utilities calculated by value iteration above to converge? That is, at what iteration of value iteration did the algorithm stop with = 0.01?   
Write your answer below and replace the None with your answer for the variable num\_convergance\_utility in the q1\_and\_2()function in answers.py.   
  
How many iterations did it take for the derived policy to stabilize? That is, at what iteration of value iteration did the policy generated by each iterations’ values stop changing? You can check this by using ascii\_grid\_policy() in your value iteration code in mdp.py.   
Write your answer below and replace the None with your answer for the variable num\_convergance\_policy in the q1\_and\_2()function in answers.py.

Utilities: \_\_\_\_\_\_\_\_ Policy: \_\_\_\_\_\_\_\_

## 3. The Big Bad (5 points)

Modify the parameters in your code so that state (2, 2) has a reward of -100 and replace the None values for the variables gridworld, EPSILON, and discount\_factor in the q3()function in answers.py and run it. Fill in the table on the right with the policy derived from the converged utilities.

Did this result in a changed policy? Replace the value for variable changed\_policy\_answer in q3() with the letter from your answer below below.

1. Yes, because the resulting policy will always take actions that have a 0.0 chance of reaching state (1, 2) with the big bad reward of -100.
2. Yes, because the resulting policy will try to ensure that we reach the goal state as quickly as possible.
3. No, because the negative reward from the state (1,2) is not enough to change the actions taken by the original policy.
4. No, because the original policy already took actions that have a 0.0 chance of reaching state (1, 2) with the big bad reward of -100.

## 4. Less Certain (5 points)

Restore the original reward structure but modify the stochasticity of the transition probabilities so that the agent achieves the desired action with a probability of 0.5 (moving 90 degrees to either side with a probability of 0.25 each). Again, replace the None values for the variables gridworld, EPSILON, and discount\_factor in the q4()function in answers.py and run it. Show the derived policy on the right.

Does this policy differ from the one in question 1? Replace the value for variable changed\_policy\_answer in the q4() with the letter for your answer below.

1. Yes, because, given the increased stochasticity, the resulting policy will always take actions that have a 0.0 chance of reaching state (1, 2) with the reward of -5.
2. Yes, because the resulting policy will try to ensure that we reach the goal state as quickly as possible to lessen the chance of hitting state (1,2) repeatedly.
3. No, because the values from value iteration for each state are sufficiently large enough such that the policy remains the same even with the increased stochasticity.
4. No, because transition probabilities have no effect on the state values given by value iteration.

## 

## 5. Heavy Discount (5 points)

After restoring the original transition probabilities, change the discount factor to 0.6 and replace the None values for the variables gridworld, EPSILON, and discount\_factor in the q5()function in answers.py. Then, fill in the resulting policy in the table on the right.

Does this policy differ from the one in question 1? Replace the value for variable changed\_policy\_answer in the q5() with the letter for your answer below.

1. Yes, because the lower the discount factor, the more you care about the immediate reward rather than the long-term rewards so the agent wants to reach the terminal state as fast as possible.
2. Yes, because the lower the discount factor, the more you care about the long-term reward rather than the immediate rewards so the agent can spend more time exploring before reaching the terminal state.
3. No, because the discount factor has no effect on the state values given by value iteration.
4. No, because the reward for the terminal state is not high enough for the lower discount factor to have an impact on the policy

## What to Submit

You should submit your version of the Python files mdp.py and answers.py, a file named homework5code.pdf with your answers and tables for the questions above. Additionally, create a readme.txt containing:

* Your name(s)
* Any noteworthy resources or people you consulting when doing your project
* Notes or warnings about what you got working, what is partially working, and what is broken