CSE 574

HW 3

Due Friday September 28 at 5pm

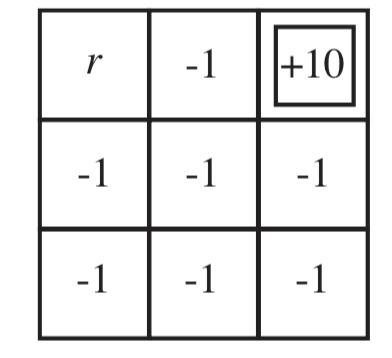
*Please submit on Blackboard. Show work for full credit. See the class policies for late HW assignments as we will strictly enforce them.*

Name:

ASU ID:

**1. Markov decision processes:**

Consider the 3 × 3 world shown in the following figure. The transition model is the following: 80% of the time the agent goes in the direction it selects; the rest of the time it moves at right angles to the intended direction i.e. If it’s intended direction is up it goes up 80% of the time up, 10% of the time goes to left and 10% of the time it goes to right.

Implement value iteration for this world for each value of r below. Use discounted rewards with a discount factor of 0.99. Show the policy obtained in each case. Explain intuitively why the value of r leads to each policy.

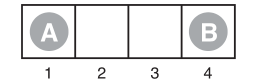
a. r = 100

b. r = −3

c. r = 0

d. r = +3

2. **Value Iteration:** Recall the zero-sum, turn-taking games of question 6 in last homework. Let the players be A and B, and let R(s) be the reward for player A in state s. (The reward for B is always equal and opposite.) Let UA(s) be the utility of state s when it is A’s turn to move in s, and let UB(s) be the utility of state s when it is B’s turn to move in s. All rewards and utilities are calculated from A’s point of view (just as in a minimax game tree).



1. Write down Bellman equations deﬁning UA (s) and UB (s).
2. Explain how to do two-player value iteration with these equations, and deﬁne a suitable termination criterion.
3. Consider the game described in Figure 5.17 on page 197. Draw the state space (rather than the game tree), showing the moves by A as solid lines and moves by B as dashed lines. Mark each state with R(s). You will ﬁnd it helpful to arrange the states (sA , sB ) on a two-dimensional grid, using sA and sB as “coordinates.”
4. Now apply two-player value iteration to solve this game, and derive the optimal policy.

3. **Finding Optimal Policies.** Dynamic Programming and Optimal Control Vol 1 #1.3 on p 54

4. **Optimal Stopping Policy in Blackjack:** Dynamic Programming and Optimal Control Vol 1 #1.4

5. **Counterfeit Coin Problem:** Dynamic Programming and Optimal Control Vol 1 #1.13

6. **Modification of the Basic Problem in DP:** Dynamic Programming and Optimal Control Vol 1 #1.14. You must prove the recursive optimality for your algorithm following an argument similar to the proof of Proposition 1.3.1 on page 25.

7.  **Proving Optimality of a Policy:** Dynamic Programming and Optimal Control Vol 1 #1.21. Use proof by induction