Assignment 5
Markov Decision Processes
Part 2: Policy Iteration

CS 3420 – Spring, 2016
Due: Wednesday, 11 May 2016

In Part 2 of this assignment, you will implement Policy Iteration to solve the large gridworld problem.

Policy Iteration

Add Policy Iteration as a solution technique in the code you wrote for Part 1. I have provided Java classes for solving systems of linear equations (written by The MathWorks, Inc. and the National Institute of Standards and Technology). They are available on BlackBoard, along with a program `TestMat.java` which shows how these classes can be used to solve linear systems. If you are using a language other than Java, similar packages are available online; just be sure to state the source in your code.

Here is Policy Iteration pseudocode, which you can use as a guide:

function POLICY-ITERATION(*mdp*) **returns** a policy
  inputs: *mdp*, an MDP with states S, actions $A(s)$, transition model $P(s' \mid s, a)$
  local variables: U, a vector of utilities for states in S, initially zero
    π, a policy vector indexed by state, initially random

  repeat
    $U \leftarrow$ POLICY-EVALUATION(π, U, mdp)
    unchanged? $\leftarrow$ true
    for each state s **in** S **do**
      if $\max_{a \in A(s)} \sum_{s'} P(s' \mid s, a)\ U[s'] > \sum_{s'} P(s' \mid s, \pi[s])\ U[s']$ **then do**
        $\pi[s] \leftarrow \operatorname{argmax}_{a \in A(s)} \sum_{s'} P(s' \mid s, a)\ U[s']$
        unchanged? $\leftarrow$ false
  until *unchanged?*
  return π

Figure 17.7 The policy iteration algorithm for calculating an optimal policy.

The command line arguments should be the same as those in Part 1. Your program should run until the policy no longer changes during the policy improvement phase of the algorithm.

NOTE: Even though there are terminal states, you may need to use a discount factor of less than 1.0, since a discount factor of 1.0 may cause the matrices involved to become singular and cause an error.