Due officially in class at the last lecture, on Wednesday December 5. Unofficially, you may hand in this project without penalty at the start of the final exam, which is on December 12 at 8am.

In this last programming project, you will do experiments with reinforcement learning. The environment to use is the 3 by 4 gridworld shown in Figure 17.1 of Russell and Norvig (page 614).

To begin, consider the case where the agent knows the transition probabilities explicitly. For this case, you should implement in Matlab the policy iteration method described in Figure 17.7. Use Matlab’s built-in features to solve simultaneous linear equations. Check that your algorithm approximately produces the numbers shown in Figure 17.3. (There are multiple ways to solve systems of linear equations in Matlab. Look for one that is efficient, i.e. that does not do full matrix inversion.)

Next, consider the true reinforcement learning case, where the agent must gradually learn the transition probabilities. Implement the Q-learning algorithm given in Figure 21.8 on page 776. Note that you must write code that simulates the environment, i.e. that generates the next state $s'$ appropriately after the agent chooses an action $a$. This simulation code will use the true transition probabilities, but the learning algorithm cannot use them. In order to make systematic experimentation feasible, your implementation should be able to simulate at least 1000 trials per second. (A trial is one run of the agent in the gridworld.)

Check that your implementation of Q-learning can indeed find the optimal policies shown in Figure 17.2 when it is initialized with Q values that are close to optimal. Then, do experiments with (i) different learning rates, (ii) different exploration strategies (epsilon-greedy versus softmax), and (iii) different initial-
izations for the Q values. Search systematically for settings of the Q-learning algorithm that can learn an optimal policy in the minimum number of trials.

For these experiments, some good metrics to report are (i) the cumulative total reward after each trial, (ii) the utility of the start state after each trial, and/or (iii) the number of trials until an optimal policy is learned. It is common for the learned policy to be optimal even though the Q values are still inaccurate. Therefore, you should measure the goodness of a learned policy using the policy evaluation routine from the policy iteration part of the project, not using the learned Q values. (Evaluate the current policy only after every $D$ trials, where you choose $D$ so that the time to evaluate policies does not slow down your code significantly.)

**What to submit.** You should hand in a printout of your code, a printout of one or two sample runs showing how it works, and a short printed report. Your report should mainly describe the design of your experiments and present their results. Use figures (for inspiration, see Figure 21.6 on page 772) and tables to present your results, and organize the report well.

Like before, for this assignment you may work either alone or in a team with one other student—your choice. If you work with someone else, you should hand in just one report and you will both get the same score for the project. Everything you submit should be printed, then stapled together and handed in at the start of lecture.