CSCI 4150 Introduction to Artificial Intelligence, Fall 1999
Assignment 8: out Thursday November 18, due midnight, evening of Friday December 3

This is the framework for Assignment 8. The complete assignment, along with support code, will be up on the web page in a day or two. I'll also have hardcopies outside my office door and in class on Tuesday November 22.

You will probably need to review the topics covered in this assignment. Inference in Bayesian belief networks is covered in Section 15.3, Bayesian classifiers are covered briefly in Section 19.6 and in handouts from class. Q-learning draws from Sections 17.1–3, but is primarily discussed in Sections 20.1–7.

1 Bayesian belief networks and classifiers

A few short written problem to calculate conditional probablilities in a Bayesian belief network and to explore Bayesian classifiers (optimal and/or naive).

2 Q-learning

For this problem, you will write the function:

```
(q-learner state prev-state prev-action reward q n)
```

to implement Q-learning using a temporal difference approach. Essentially, you will implement the algorithm from Figure 20.12 from our text.

This function will be given: the (current) state, the previous state and action (that took us to this state), the reward received in the previous state, a table of Q values, and a table of action-state frequencies. It must return an action (see below).

This learning program will operate in the grid worlds such as the three by four world used as an illustration in Chapters 17 and 20. The agent is located in a particular cell and can execute one of four actions: north, south, east, and west. The agent is most likely to move in the desired direction; however, there is some chance that the agent will move either to the right, left, or backwards relative to the desired direction. (I am generalizing our text's version of this problem.) The program will continue from state to state until it reaches a terminal state, which will have either a 1.0 or -1.0 reward. All other states will receive some small negative reward (a small cost for each step in the grid).

I will provide support code to:

- simulate the world
- implement an action-state table data type
- run the learning process, i.e. set up the tables and the world, call your `q-learner` function, abstract your policy from your table of Q values, and perhaps provide some statistics on how you're doing.

In the process of implementing the `q-learner` function, you will need to figure out what functions to use for:

- alpha — the learning rate
- f — the exploration function

You may also need to address how you resolve ties in exploration function values.