Reinforcement Learning - Active Agent in NxN Grid world

Introduction:

For HW3, you will implement an active agent that employs reinforcement learning (RL) that employs an "exploitation/exploration" approach (as discussed in class, and in the text) to discover a policy in an unexplored grid world. The program will run on well.cs.ucr.edu and be turned in with source and readme to ilearn.

Details:

The world to be explored will consist of m (number) of +10 rewards and an equal number of -10 penalties randomly placed through an nxn grid world, their positions not known to the agent. The world will also have n randomly placed 1x1 size grid block obstacles, that do not occupy the same space as the rewards/penalties. n and m are inputs to the program, input at the command line, as in:

bash-$> **a.out n m**

For full credit, the RL-agent will utilize a discount term for its value calculations and a learning rate that decreases over time. These can be hard-coded. A random policy will start and a random start state (fixed for that world) will be where the agent spawns. Like the "4x3" grid world (examples in the book and videos), the agent will act with the same probability with stochastic actions (i.e. 0.8 to follow the desired action, but 0.1 it ends up heading left (of the action), and 0.1 it ends up to the right of the intended action; also once the direction is defined by this distribution, no action occurs if it is attempting to move through an obstacle for the probability-selected course).

The program should terminate when the value function improvement falls below a set threshold; or when a maximum number of iterations is met. Threshold and max_iterations may be hard-coded.

Output:

While I am in discussion with the TA about a more beautified "graphics" output, for now, we will keep it simple. That is, upon termination, the program will output its policy and value function, in ascii to the terminal, following this 5x5 example with m = 1:

```
x  v  <  x  v
>  +  <  <  ^
>  ^  x  <  -
x  ^  x  ^  v
>  s  <  ^  <
```

Legend:
<>^v = {up,down,left,right}
+ = Reward +10 // terminal state
- = Penalty -10 // terminal state
x = Obstacle
s = Agent Start // no action shown

and the values are the V(s) for non-obstacle states as:

```
x  #  #  x  #
#  #  #  #  #
#  #  x  #  #
x  #  x  #  #
#  #  #  #  #
```

Obstacles are x again in the value function output.