

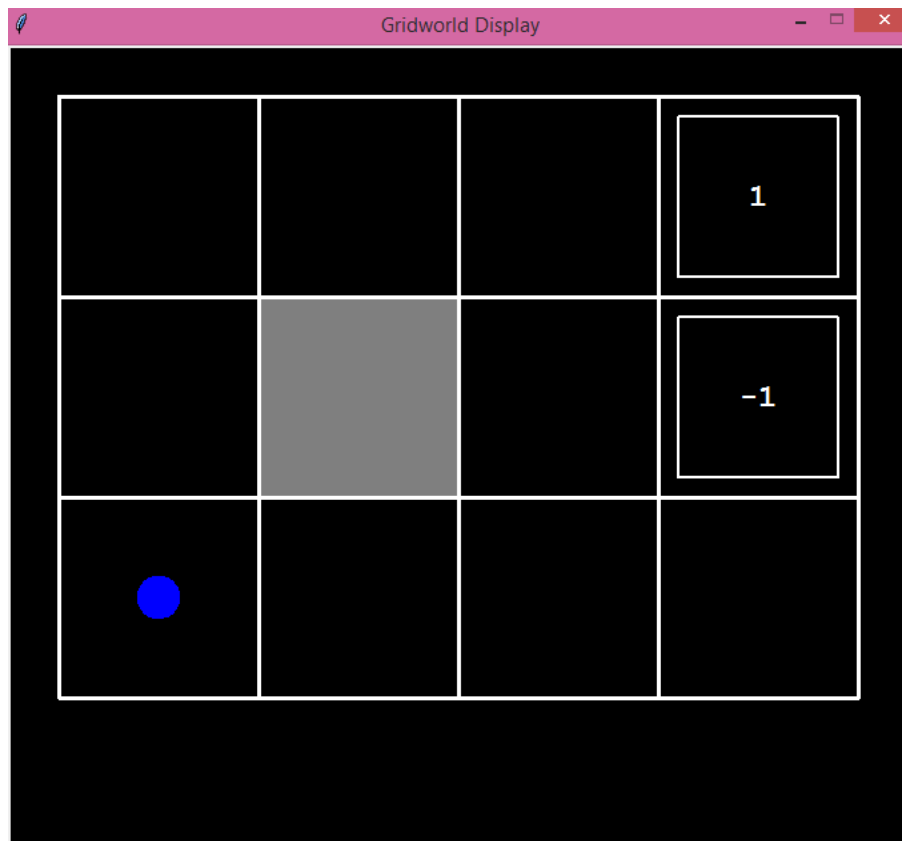
## Part 3

### *MDP and RL Analysis*

1. Part 2-Q1 - Run value iteration on default BookGrid for 5, 10, 15, 20, 30, 50 iterations. Show screenshots and comment at which point did the values of states and actions converge

To run Gridworld in manual control mode, which uses the arrow keys:

```
python gridworld.py -m
```



```
C:\Users\HP\Desktop\PUtry2>python gridworld.py -m
## Disabling Agents in Manual Mode (-m) ##

RUNNING 1 EPISODES

BEGINNING EPISODE: 1

Started in state: (0, 0)
Took action: west
Ended in state: (0, 0)
Got reward: 0.0

Started in state: (0, 0)
Took action: north
Ended in state: (0, 0)
Got reward: 0.0

Started in state: (0, 0)
Took action: north
Ended in state: (0, 1)
Got reward: 0.0

Started in state: (0, 1)
Took action: north
Ended in state: (0, 2)
Got reward: 0.0

Started in state: (0, 2)
Took action: east
Ended in state: (1, 2)
Got reward: 0.0

Started in state: (1, 2)
Took action: east
Ended in state: (2, 2)
Got reward: 0.0

Started in state: (2, 2)
Took action: east
Ended in state: (3, 2)
Got reward: 0.0

Started in state: (3, 2)
Took action: exit
Ended in state: TERMINAL_STATE
Got reward: 1

EPISODE 1 COMPLETE: RETURN WAS 0.478296900000000014

AVERAGE RETURNS FROM START STATE: 0.478296900000000014
```

## Question 1 : Value Iteration

Testing the implementation :

```
python autograder.py -q q1
```

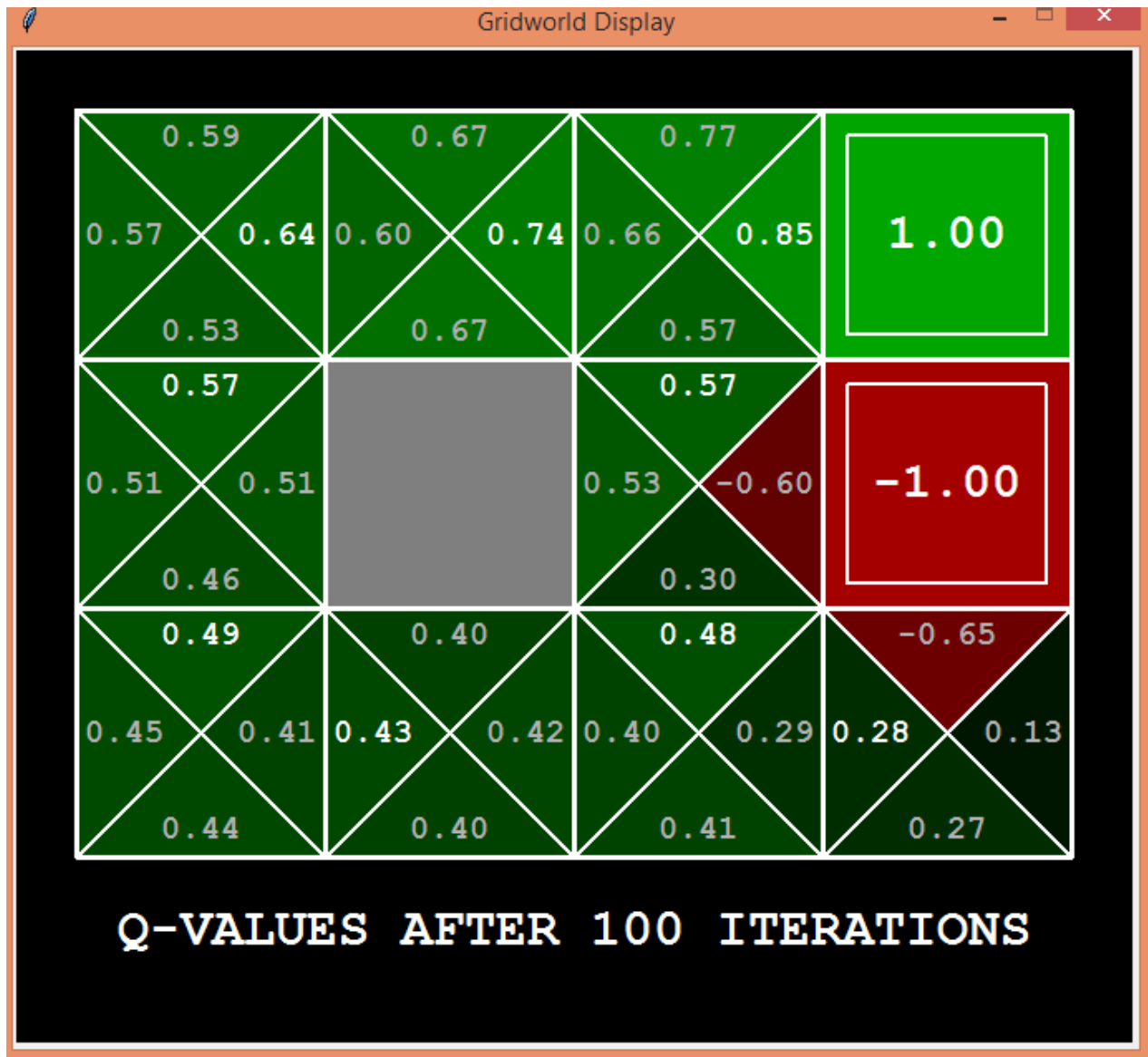
```
Question q1
=====
*** PASS: test_cases\q1\1-tinygrid.test
```

Implementation for value iteration is successful.

The following command loads your ValueIterationAgent, which will compute a policy and execute it 10 times. Press a key to cycle through values, Q-values, and the simulation. You should find that the value of the start state ( $V(\text{start})$ , which you can read off of the GUI) and the empirical resulting average reward (printed after the 10 rounds of execution finish) are quite close.

```
python gridworld.py -a value -i 100 -k 10
```

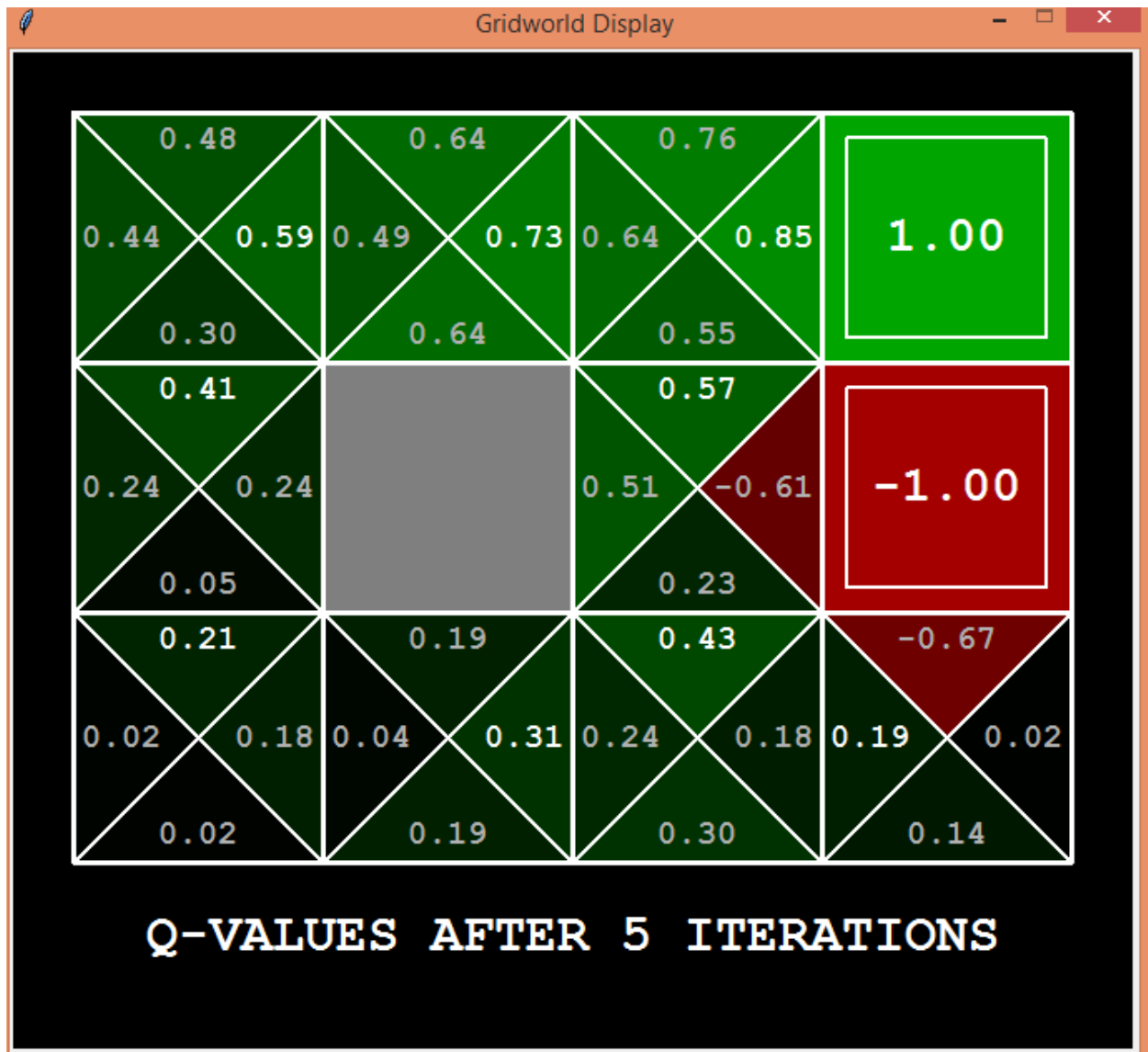




Running value iteration for 5 iterations should give you this output:

```
python gridworld.py -a value -i 5
```



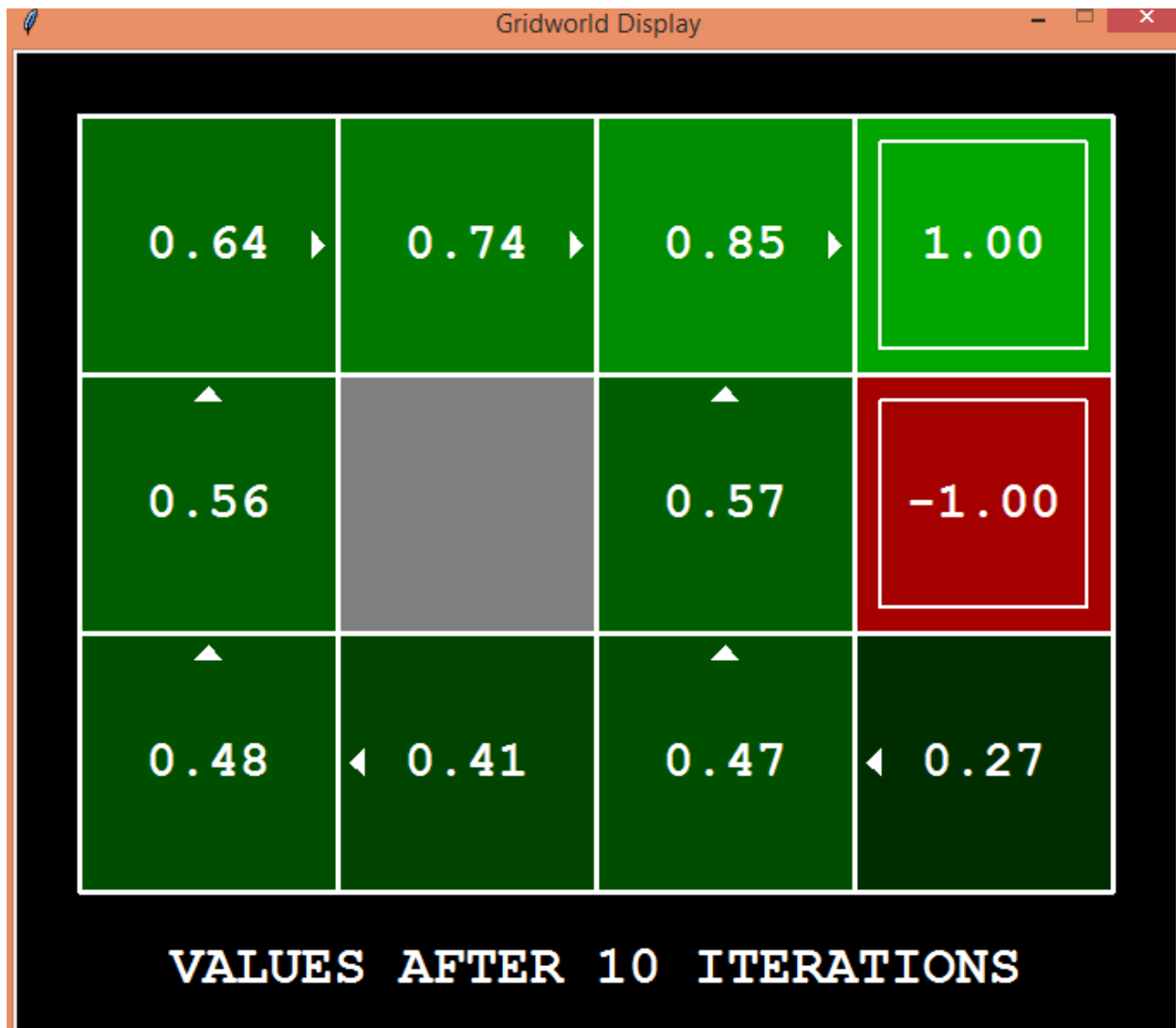


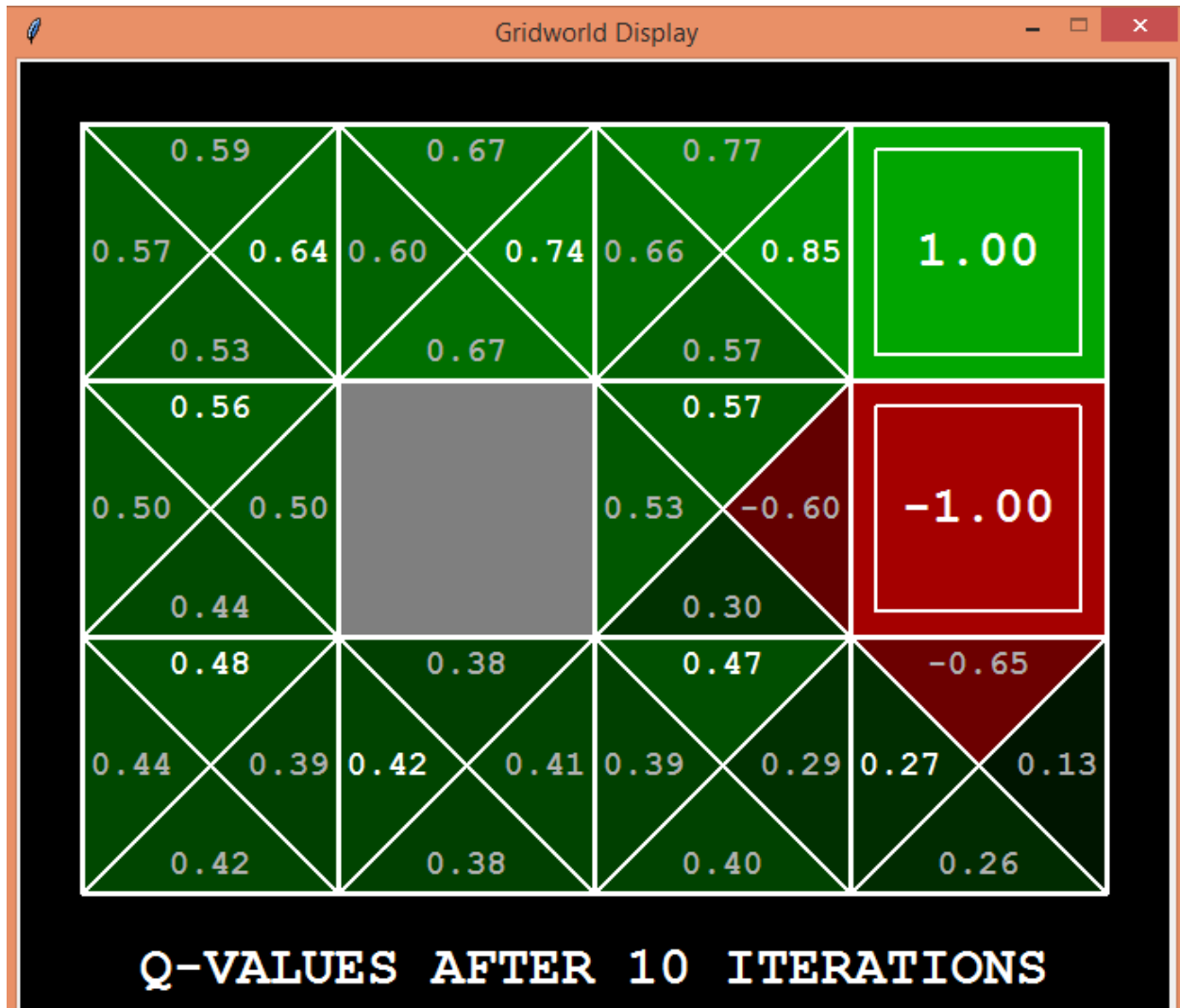


```
C:\Users\HP\Desktop\PUtry2>python gridworld.py -a value -i 5
RUNNING 1 EPISODES
BEGINNING EPISODE: 1
Started in state: (0, 0)
Took action: north
Ended in state: (0, 1)
Got reward: 0.0
Started in state: (0, 1)
Took action: north
Ended in state: (0, 2)
Got reward: 0.0
Started in state: (0, 2)
Took action: east
Ended in state: (1, 2)
Got reward: 0.0
Started in state: (1, 2)
Took action: east
Ended in state: (1, 2)
Got reward: 0.0
Started in state: (1, 2)
Took action: east
Ended in state: (2, 2)
Got reward: 0.0
Started in state: (2, 2)
Took action: east
Ended in state: (3, 2)
Got reward: 0.0
Started in state: (3, 2)
Took action: exit
Ended in state: TERMINAL_STATE
Got reward: 1
EPISODE 1 COMPLETE: RETURN WAS 0.5314410000000002
AVERAGE RETURNS FROM START STATE: 0.5314410000000002
C:\Users\HP\Desktop\PUtry2>
```

Running value iteration for 10 iterations should give you this output:

```
python gridworld.py -a value -i 10
```



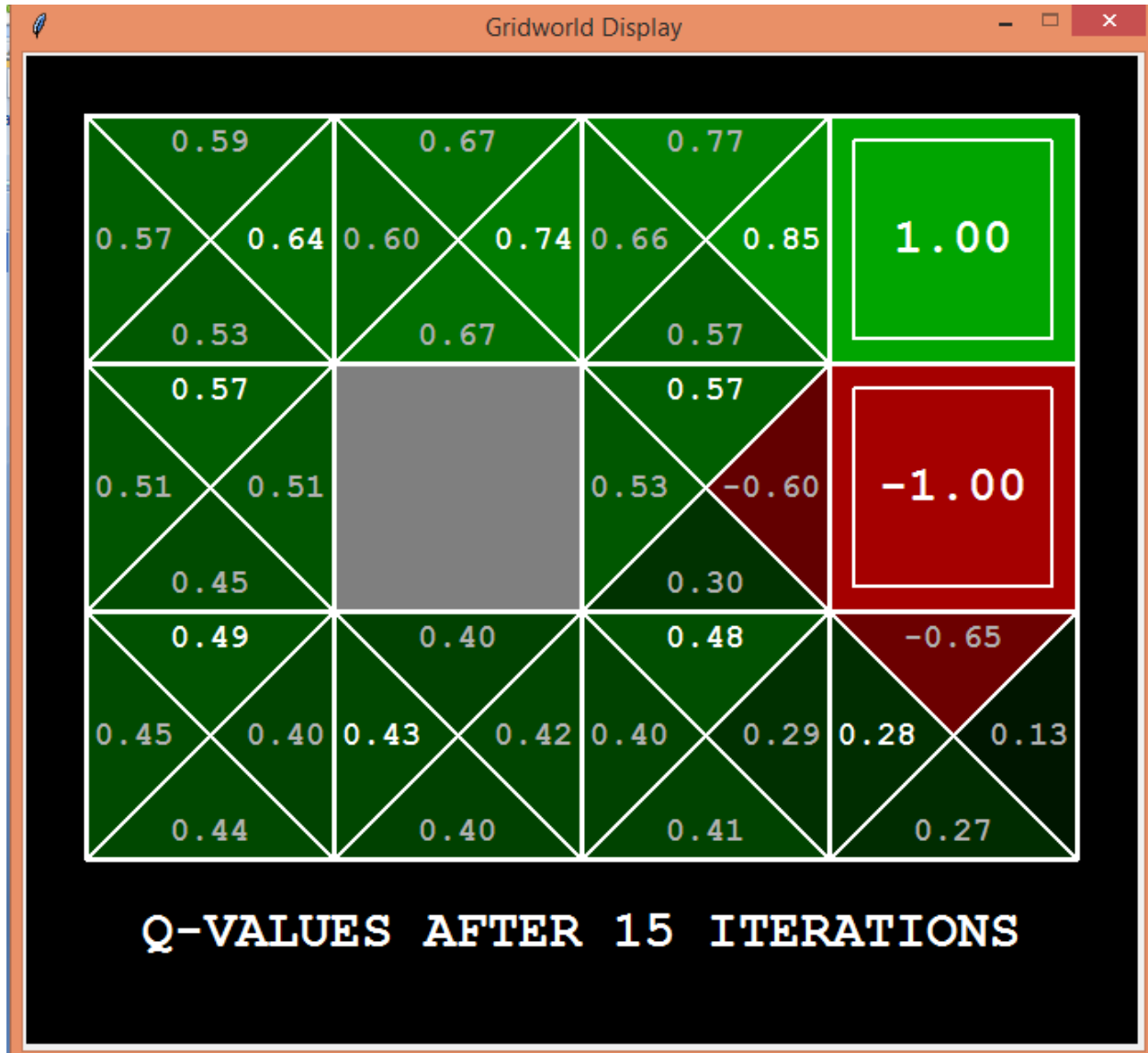


```
C:\Users\HP\Desktop\PUtry2>python gridworld.py -a value -i 10
RUNNING 1 EPISODES
BEGINNING EPISODE: 1
Started in state: (0, 0)
Took action: north
Ended in state: (0, 1)
Got reward: 0.0
Started in state: (0, 1)
Took action: north
Ended in state: (0, 2)
Got reward: 0.0
Started in state: (0, 2)
Took action: east
Ended in state: (1, 2)
Got reward: 0.0
Started in state: (1, 2)
Took action: east
Ended in state: (2, 2)
Got reward: 0.0
Started in state: (2, 2)
Took action: east
Ended in state: (3, 2)
Got reward: 0.0
Started in state: (3, 2)
Took action: exit
Ended in state: TERMINAL_STATE
Got reward: 1
EPISODE 1 COMPLETE: RETURN WAS 0.5904900000000002
AVERAGE RETURNS FROM START STATE: 0.5904900000000002
```

Running value iteration for 15 iterations should give you this output:

```
python gridworld.py -a value -i 15
```





```
Command Prompt

Started in state: (0, 0)
Took action: north
Ended in state: (0, 1)
Got reward: 0.0

Started in state: (0, 1)
Took action: north
Ended in state: (0, 2)
Got reward: 0.0

Started in state: (0, 2)
Took action: east
Ended in state: (0, 2)
Got reward: 0.0

Started in state: (0, 2)
Took action: east
Ended in state: (0, 2)
Got reward: 0.0

Started in state: (0, 2)
Took action: east
Ended in state: (1, 2)
Got reward: 0.0

Started in state: (1, 2)
Took action: east
Ended in state: (1, 2)
Got reward: 0.0

Started in state: (1, 2)
Took action: east
Ended in state: (1, 2)
Got reward: 0.0

Started in state: (1, 2)
Took action: east
Ended in state: (2, 2)
Got reward: 0.0

Started in state: (2, 2)
Took action: east
Ended in state: (3, 2)
Got reward: 0.0

Started in state: (3, 2)
Took action: exit
Ended in state: TERMINAL_STATE
Got reward: 1

EPISODE 1 COMPLETE: RETURN WAS 0.38742048900000015

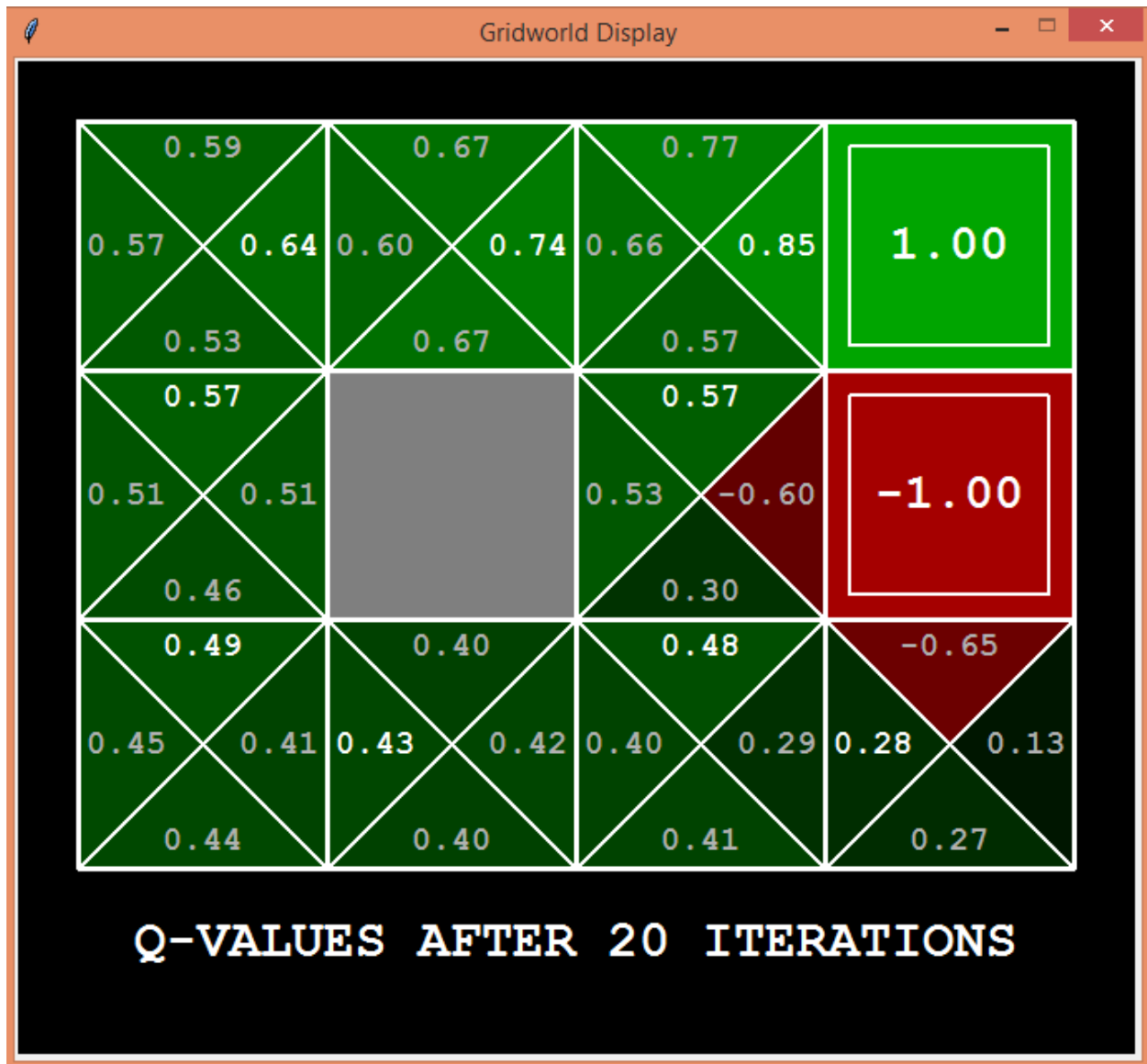
AVERAGE RETURNS FROM START STATE: 0.38742048900000015
```

Running value iteration for 20 iterations should give you this output:

```
python gridworld.py -a value -i 20
```







```
RUNNING 1 EPISODES
BEGINNING EPISODE: 1

Started in state: (0, 0)
Took action: north
Ended in state: (0, 1)
Got reward: 0.0

Started in state: (0, 1)
Took action: north
Ended in state: (0, 1)
Got reward: 0.0

Started in state: (0, 1)
Took action: north
Ended in state: (0, 2)
Got reward: 0.0

Started in state: (0, 2)
Took action: east
Ended in state: (0, 2)
Got reward: 0.0

Started in state: (0, 2)
Took action: east
Ended in state: (1, 2)
Got reward: 0.0

Started in state: (1, 2)
Took action: east
Ended in state: (1, 2)
Got reward: 0.0

Started in state: (1, 2)
Took action: east
Ended in state: (2, 2)
Got reward: 0.0

Started in state: (2, 2)
Took action: east
Ended in state: (3, 2)
Got reward: 0.0

Started in state: (3, 2)
Took action: exit
Ended in state: TERMINAL_STATE
Got reward: 1

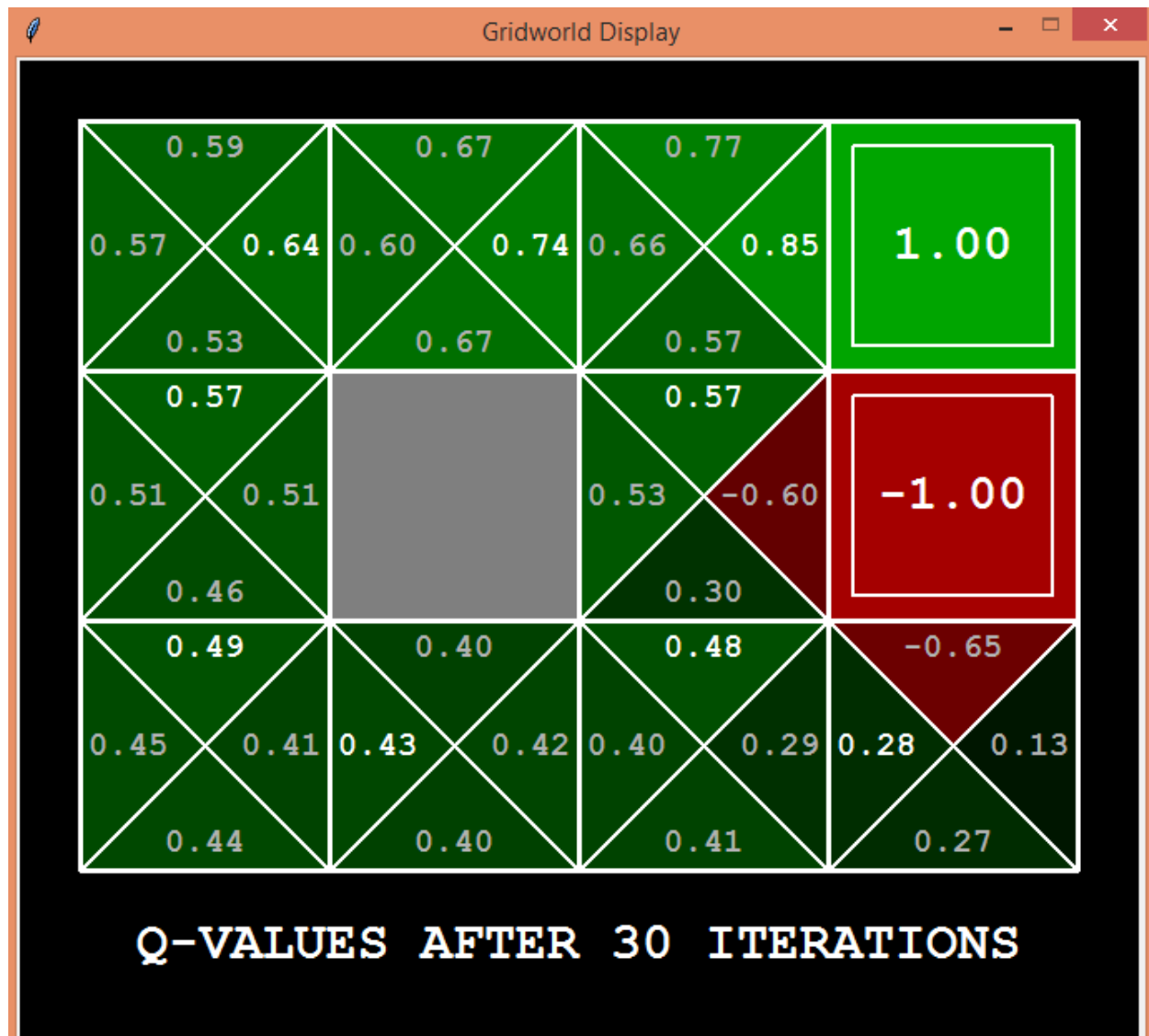
EPISODE 1 COMPLETE: RETURN WAS 0.43046721000000016

AVERAGE RETURNS FROM START STATE: 0.43046721000000016
```

Running value iteration for 30 iterations should give you this output:

```
python gridworld.py -a value -i 30
```





```
Got reward: 1
EPISODE 1 COMPLETE: RETURN WAS 0.430467210000000016

AVERAGE RETURNS FROM START STATE: 0.430467210000000016

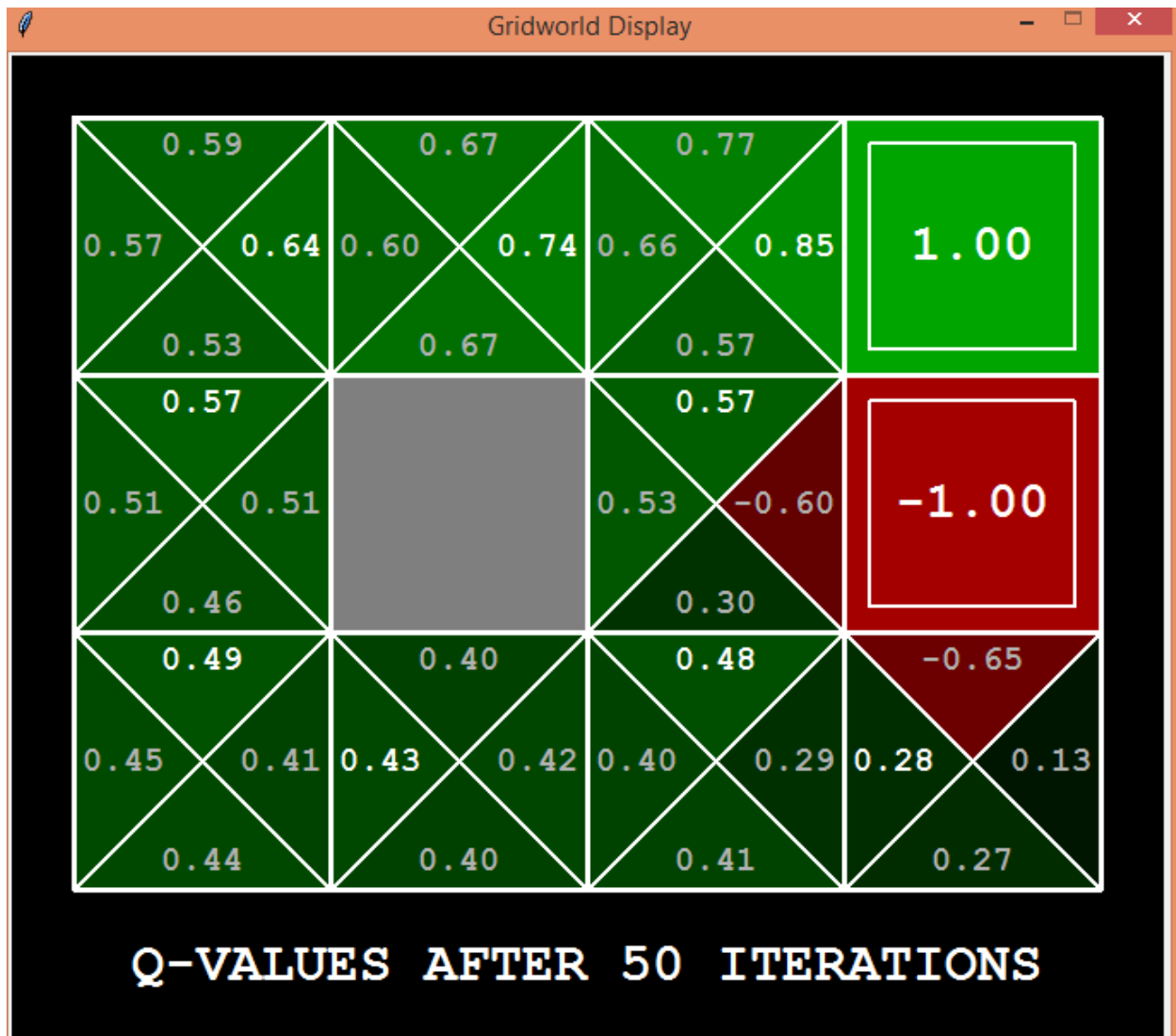
C:\Users\HP\Desktop\PUtry2>python gridworld.py -a value -i 30
RUNNING 1 EPISODES
BEGINNING EPISODE: 1
Started in state: (0, 0)
Took action: north
Ended in state: (0, 1)
Got reward: 0.0
Started in state: (0, 1)
Took action: north
Ended in state: (0, 2)
Got reward: 0.0
Started in state: (0, 2)
Took action: east
Ended in state: (1, 2)
Got reward: 0.0
Started in state: (1, 2)
Took action: east
Ended in state: (1, 2)
Got reward: 0.0
Started in state: (1, 2)
Took action: east
Ended in state: (2, 2)
Got reward: 0.0
Started in state: (2, 2)
Took action: east
Ended in state: (3, 2)
Got reward: 0.0
Started in state: (3, 2)
Took action: exit
Ended in state: TERMINAL_STATE
Got reward: 1
EPISODE 1 COMPLETE: RETURN WAS 0.53144100000000002

AVERAGE RETURNS FROM START STATE: 0.53144100000000002
```

Running value iteration for 50 iterations should give you this output:

```
python gridworld.py -a value -i 50
```





```
C:\ Command Prompt
Started in state: (0, 0)
Took action: north
Ended in state: (1, 0)
Got reward: 0.0

Started in state: (1, 0)
Took action: west
Ended in state: (0, 0)
Got reward: 0.0

Started in state: (0, 0)
Took action: north
Ended in state: (1, 0)
Got reward: 0.0

Started in state: (1, 0)
Took action: west
Ended in state: (0, 0)
Got reward: 0.0

Started in state: (0, 0)
Took action: north
Ended in state: (0, 1)
Got reward: 0.0

Started in state: (0, 1)
Took action: north
Ended in state: (0, 2)
Got reward: 0.0

Started in state: (0, 2)
Took action: east
Ended in state: (1, 2)
Got reward: 0.0

Started in state: (1, 2)
Took action: east
Ended in state: (2, 2)
Got reward: 0.0

Started in state: (2, 2)
Took action: east
Ended in state: (3, 2)
Got reward: 0.0

Started in state: (3, 2)
Took action: exit
Ended in state: TERMINAL_STATE
Got reward: 1

EPISODE 1 COMPLETE: RETURN WAS 0.38742048900000015

AVERAGE RETURNS FROM START STATE: 0.38742048900000015
```



In the following figure I've merged the values of all the iterations to get the idea about the values of states and actions converge.

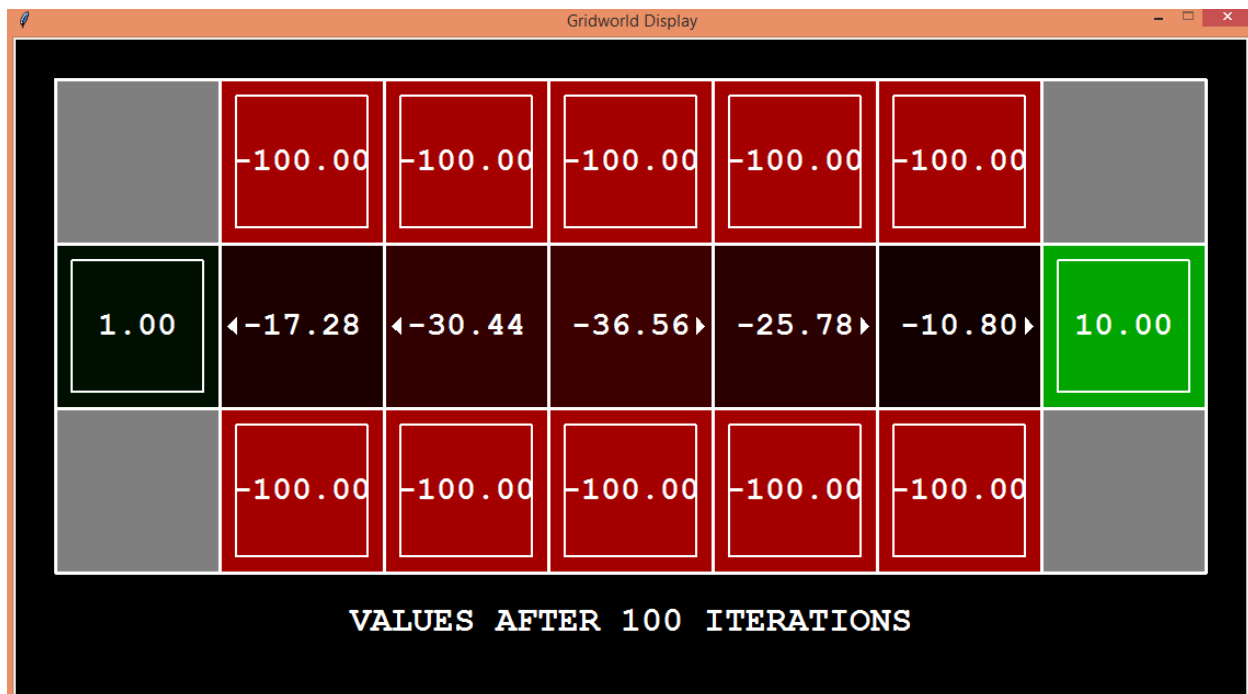


From the above figure we can see that after the first iteration in every iteration the values are getting converged. Since it is a type of MDP and RL, it keeps the history of states and actions of Q learning and some of the values are the same for the result of the next iteration. That is how an agent learns from his percept history and uses it for further moves.

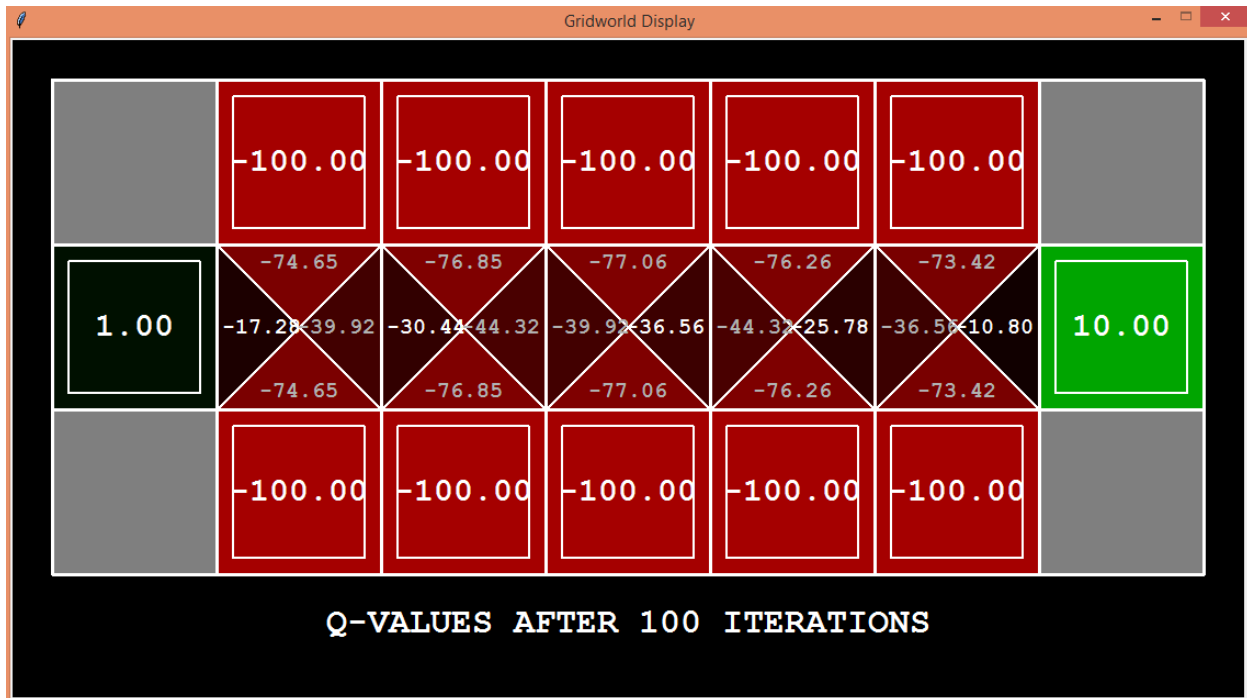
2. Part 2-Q1 - Run value iteration on BridgeGrid for 10 iterations and show the result. Modify discount rate from the default 0.9 to 0.1, and run again for 10 iterations. Comment if/why results differ.

## Bridge Crossing Analysis

```
python gridworld.py -a value -i 100 -g BridgeGrid --discount 0.9 --noise 0.2
```



BridgeGrid is a grid world map with the a low-reward terminal state and a high-reward terminal state separated by a narrow “bridge”, on either side of which is a chasm of high negative reward. The agent starts near the low-reward state. With the default discount of 0.9 and the default noise of 0.2, the optimal policy does not cross the bridge.



```
C:\Users\HP\Desktop\PUtry2>python gridworld.py -a value -i 100 -g BridgeGrid --discount 0.9 --noise 0.2
```

```
RUNNING 1 EPISODES
```

```
BEGINNING EPISODE: 1
```

```
Started in state: (1, 1)
```

```
Took action: west
```

```
Ended in state: (0, 1)
```

```
Got reward: 0.0
```

```
Started in state: (0, 1)
```

```
Took action: exit
```

```
Ended in state: TERMINAL_STATE
```

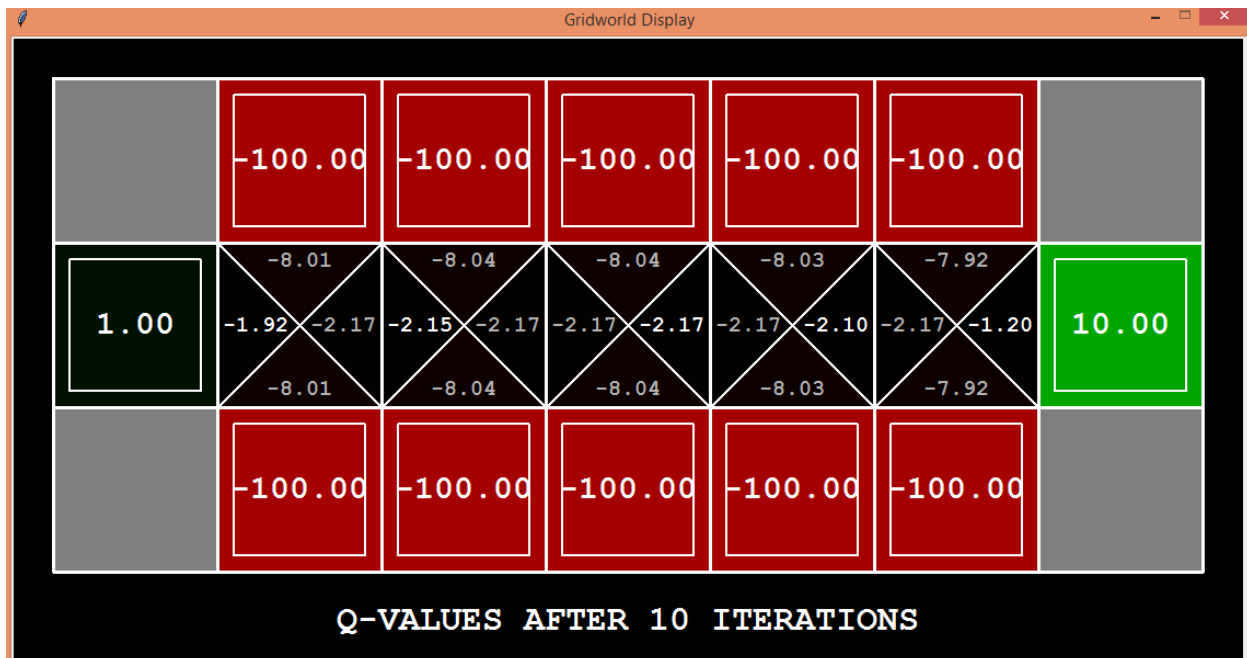
```
Got reward: 1
```

```
EPISODE 1 COMPLETE: RETURN WAS 0.9
```

```
AVERAGE RETURNS FROM START STATE: 0.9
```

Now, changing the discount rate from the default 0.9 to 0.1 and running it for 10 iterations.

```
python gridworld.py -a value -i 10 -g BridgeGrid --discount 0.1 --noise 0.2
```

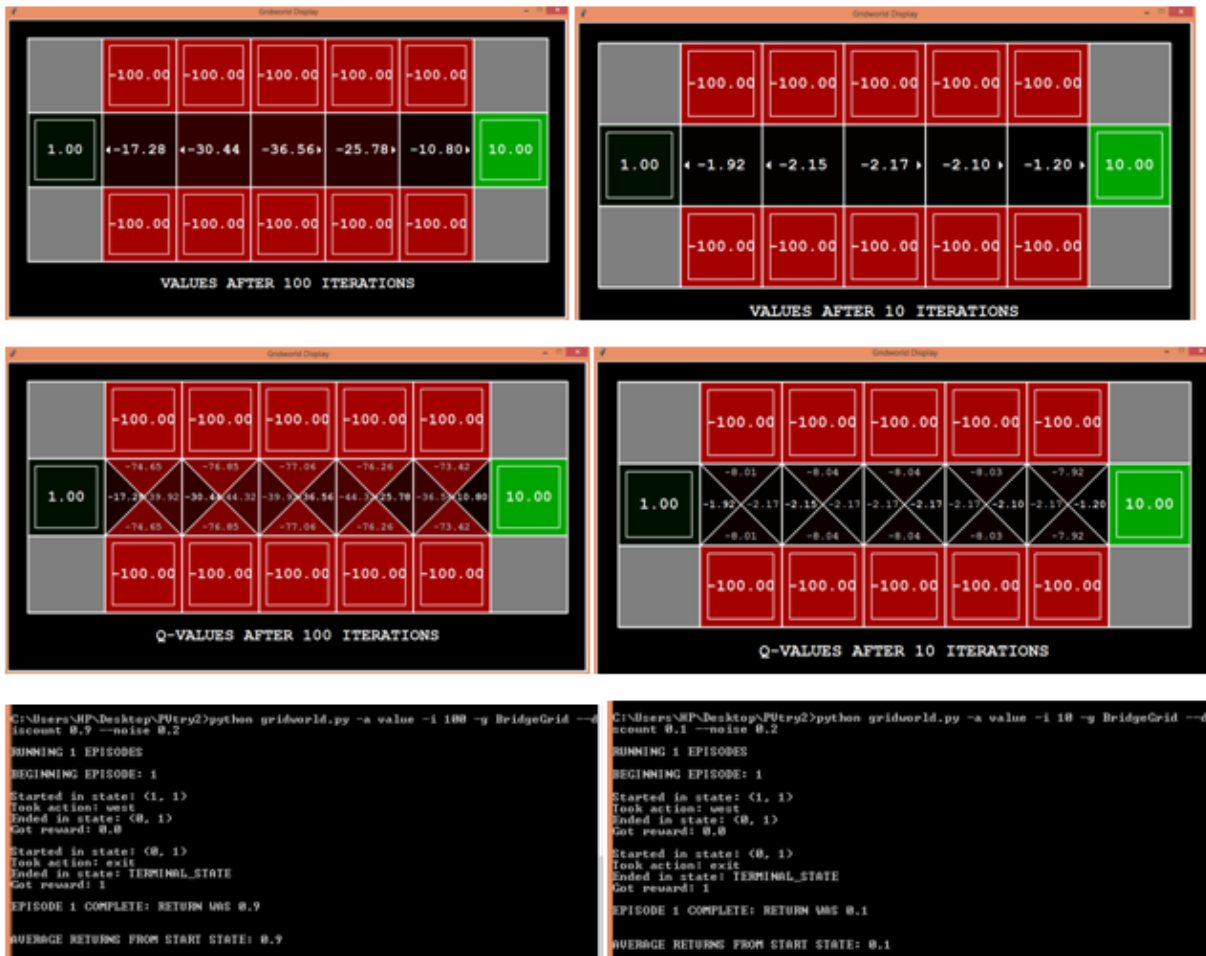


```
C:\Users\HP\Desktop\PUtry2>python gridworld.py -a value -i 10 -g BridgeGrid --di
scount 0.1 --noise 0.2

RUNNING 1 EPISODES
BEGINNING EPISODE: 1
Started in state: (1, 1)
Took action: west
Ended in state: (0, 1)
Got reward: 0.0
Started in state: (0, 1)
Took action: exit
Ended in state: TERMINAL_STATE
Got reward: 1
EPISODE 1 COMPLETE: RETURN WAS 0.1

AVERAGE RETURNS FROM START STATE: 0.1
```

Following figure is a comparison of 100 iteration command and 10 iteration command:

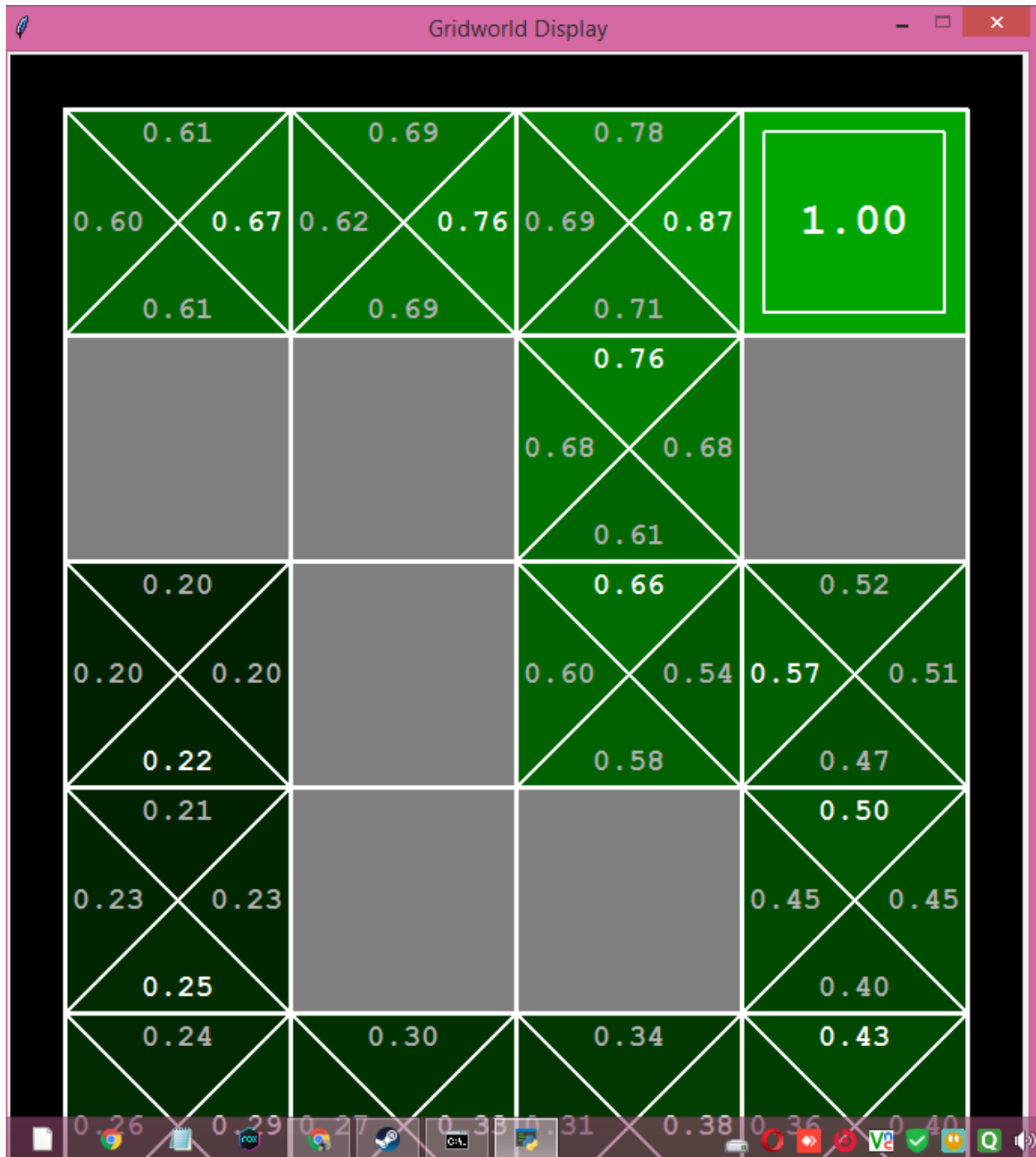


From the comparison we can state that both the results vary a lot. The values of states and actions are not mapping to each other. Also, The time required for the episodes varies a lot. The average returning for 100 iterations was 0.9 and for 10 iteration it came out to be 0.1. This huge difference is seen because we have changed the discount rate and the number of iterations with a huge difference. As the discount rate, noise, values changes the results get's varied.

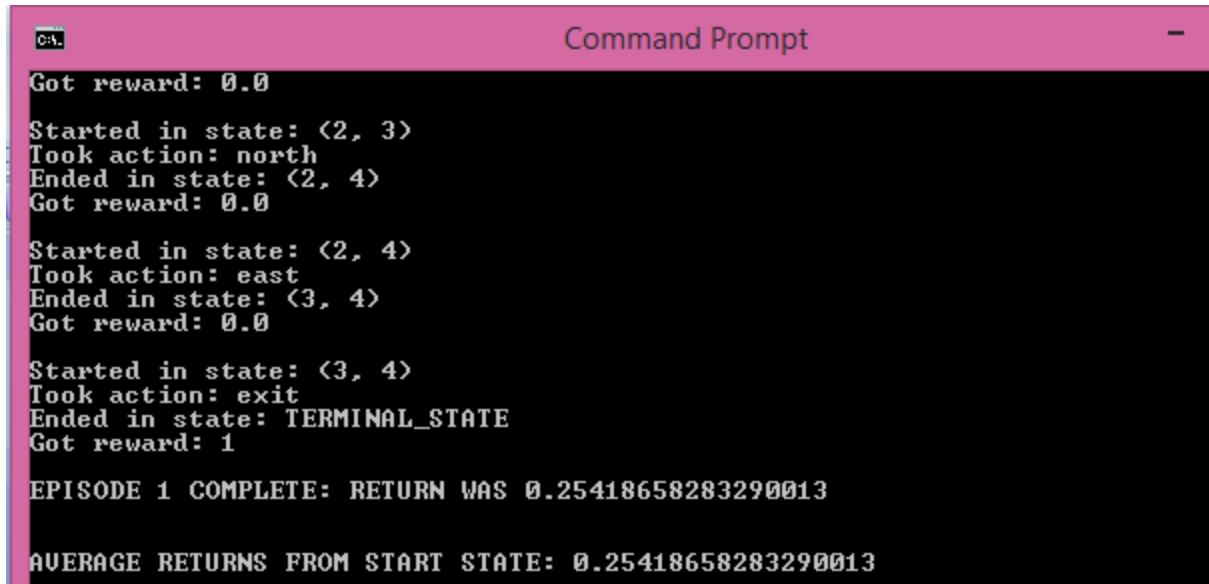
Part 2 – Q4 - Run value iteration on MazeGrid for 100 iterations. Show final screenshot. Then run asynchronous value iteration you just implemented on the same grid for 100 iterations. Show screenshot. Explain the difference. Run it again for 1000 iterations and compare the results.

```
python gridworld.py -a value -i 100 -g MazeGrid
```







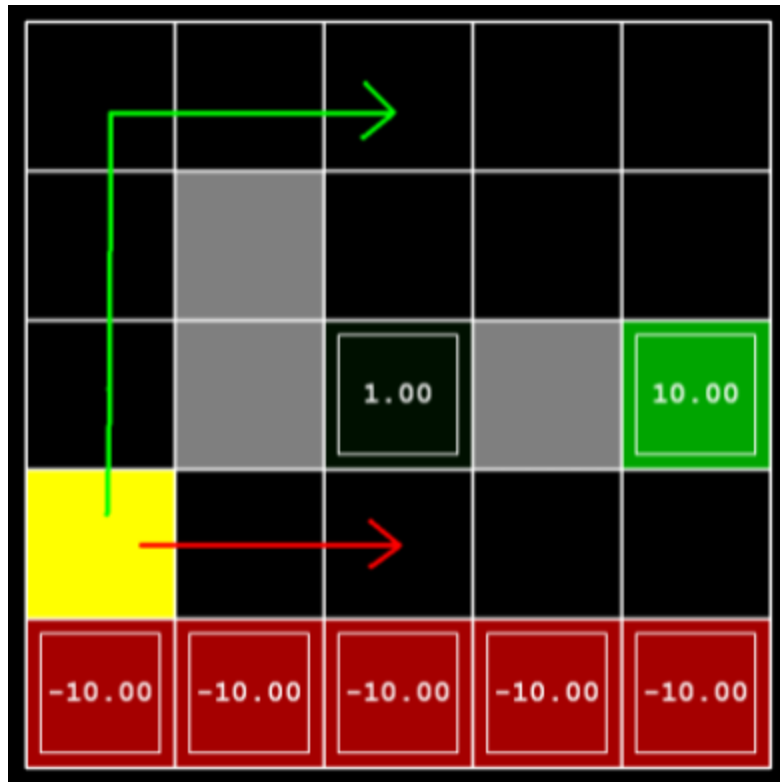


```
Got reward: 0.0
Started in state: (2, 3)
Took action: north
Ended in state: (2, 4)
Got reward: 0.0
Started in state: (2, 4)
Took action: east
Ended in state: (3, 4)
Got reward: 0.0
Started in state: (3, 4)
Took action: exit
Ended in state: TERMINAL_STATE
Got reward: 1
EPISODE 1 COMPLETE: RETURN WAS 0.25418658283290013
AVERAGE RETURNS FROM START STATE: 0.25418658283290013
```

## Policies

Checking the code implementation for Q. 3:

```
python autograder.py -q q3
```



```
C:\Users\HP\Desktop\PUtry2>python autograder.py -q q3
C:\Users\HP\Desktop\PUtry2\autograder.py:17: DeprecationWarning: the imp
is deprecated in favour of importlib; see the module's documentation for
tive uses
    import imp
Starting on 3-7 at 14:54:33

Question q3
=====

*** PASS: test_cases\q3\1-question-3.1.test
```

The output proves that Q.3 is successfully implemented.

Now, Checking implementation for Q. 4:

```
python autograder.py -q q4
```

```
C:\Users\HP\Desktop\PUtry2>python autograder.py -q q4
C:\Users\HP\Desktop\PUtry2>autograder.py:17: DeprecationWarning: the imp module
is deprecated in favour of importlib; see the module's documentation for a
replacement
  import imp
Starting on 3-7 at 14:56:26

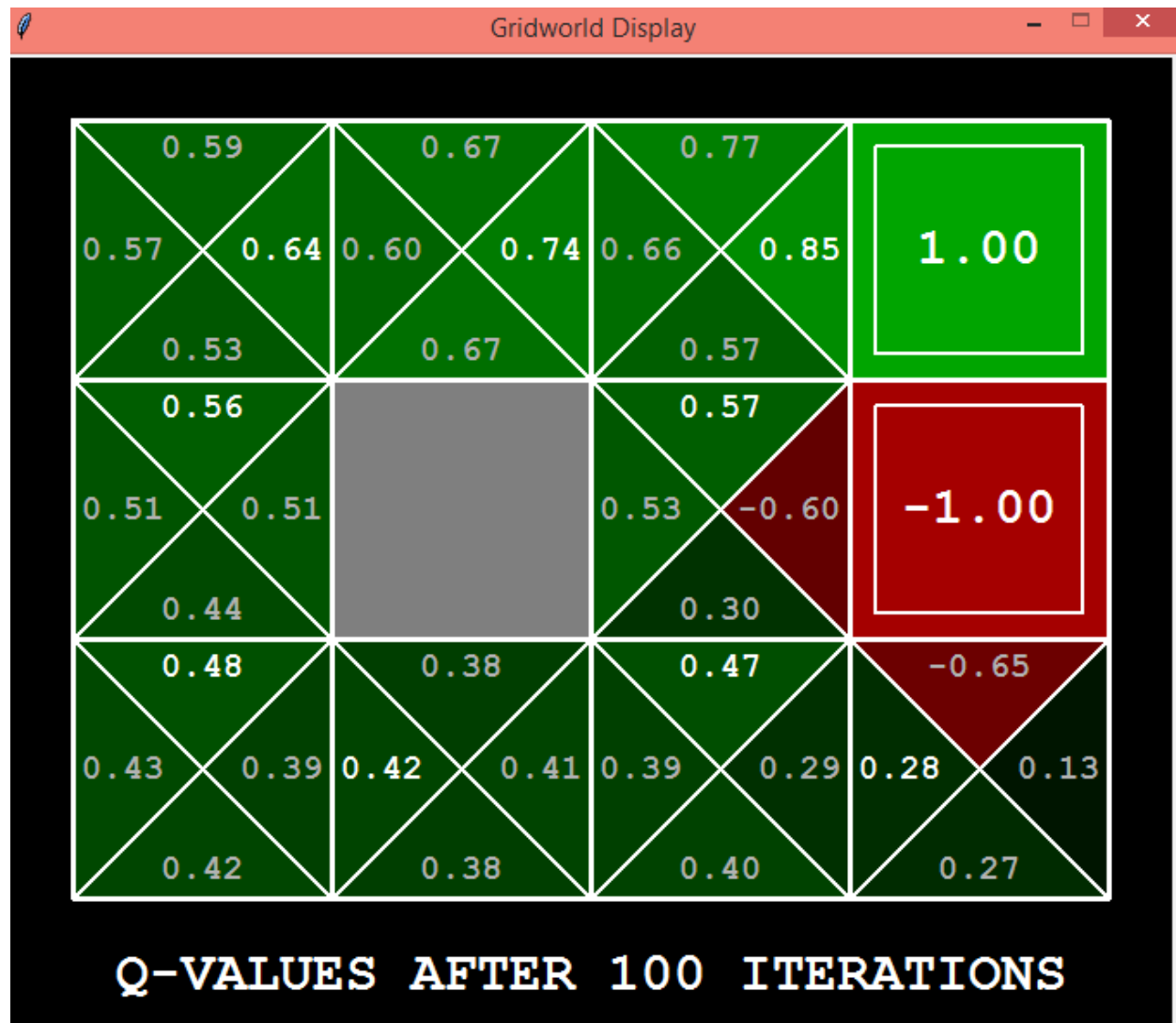
Question q4
=====
*** PASS: test_cases\q4\1-tinygrid.test
```

The output proves that Q.4 is successfully implemented.

## Asynchronous Value Iteration

```
python gridworld.py -a asynchvalue -i 100 -k 10
```





```
Command Prompt

EPISODE 9 COMPLETE: RETURN WAS 0.59049000000000002
BEGINNING EPISODE: 10
Started in state: (0, 0)
Took action: north
Ended in state: (1, 0)
Got reward: 0.0
Started in state: (1, 0)
Took action: west
Ended in state: (0, 0)
Got reward: 0.0
Started in state: (0, 0)
Took action: north
Ended in state: (0, 1)
Got reward: 0.0
Started in state: (0, 1)
Took action: north
Ended in state: (0, 1)
Got reward: 0.0
Started in state: (0, 1)
Took action: north
Ended in state: (0, 2)
Got reward: 0.0
Started in state: (0, 2)
Took action: east
Ended in state: (1, 2)
Got reward: 0.0
Started in state: (1, 2)
Took action: east
Ended in state: (2, 2)
Got reward: 0.0
Started in state: (2, 2)
Took action: east
Ended in state: (3, 2)
Got reward: 0.0
Started in state: (3, 2)
Took action: exit
Ended in state: TERMINAL_STATE
Got reward: 1
EPISODE 10 COMPLETE: RETURN WAS 0.430467210000000016
AVERAGE RETURNS FROM START STATE: 0.5136932916090001
```

Let's check the working of implementation for Q. 5:

```
python autograder.py -q q5
```

```
C:\Users\HP\Desktop\PUtry2>python autograder.py -q q5
C:\Users\HP\Desktop\PUtry2>autograder.py:17: DeprecationWarning: the imp module
is deprecated in favour of importlib; see the module's documentation for alte
rnative uses
  import imp
Starting on 3-7 at 15:01:49

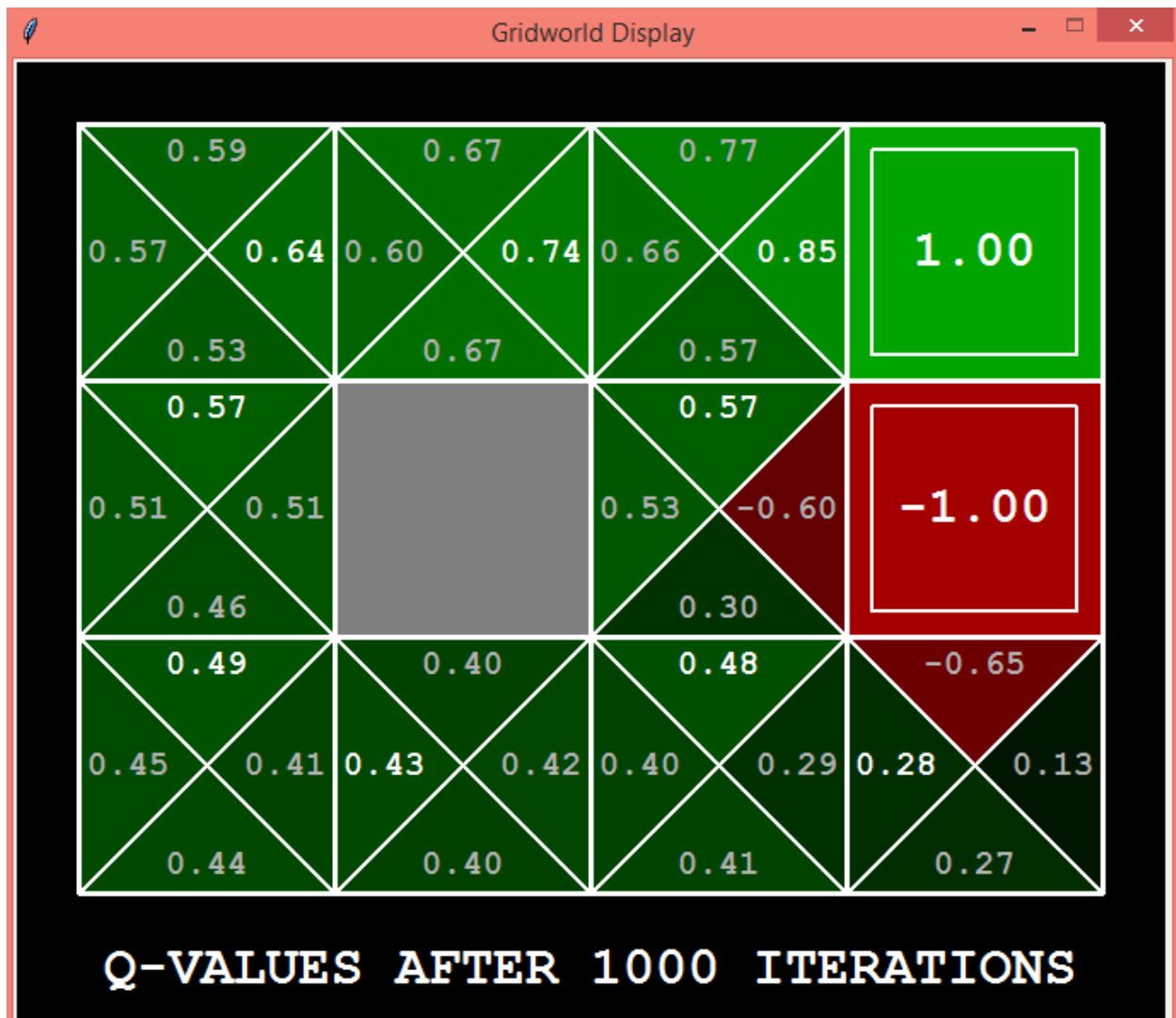
Question q5
=====
*** PASS: test_cases\q5\1-tinygrid.test
```

The output implies that implementation is successful.

```
python gridworld.py -a priosweepvalue -i 1000
```







```
Command Prompt

Started in state: (0, 0)
Took action: north
Ended in state: (0, 0)
Got reward: 0.0

Started in state: (0, 0)
Took action: north
Ended in state: (0, 0)
Got reward: 0.0

Started in state: (0, 0)
Took action: north
Ended in state: (0, 1)
Got reward: 0.0

Started in state: (0, 1)
Took action: north
Ended in state: (0, 2)
Got reward: 0.0

Started in state: (0, 2)
Took action: east
Ended in state: (1, 2)
Got reward: 0.0

Started in state: (1, 2)
Took action: east
Ended in state: (1, 2)
Got reward: 0.0

Started in state: (1, 2)
Took action: east
Ended in state: (2, 2)
Got reward: 0.0

Started in state: (2, 2)
Took action: east
Ended in state: (2, 2)
Got reward: 0.0

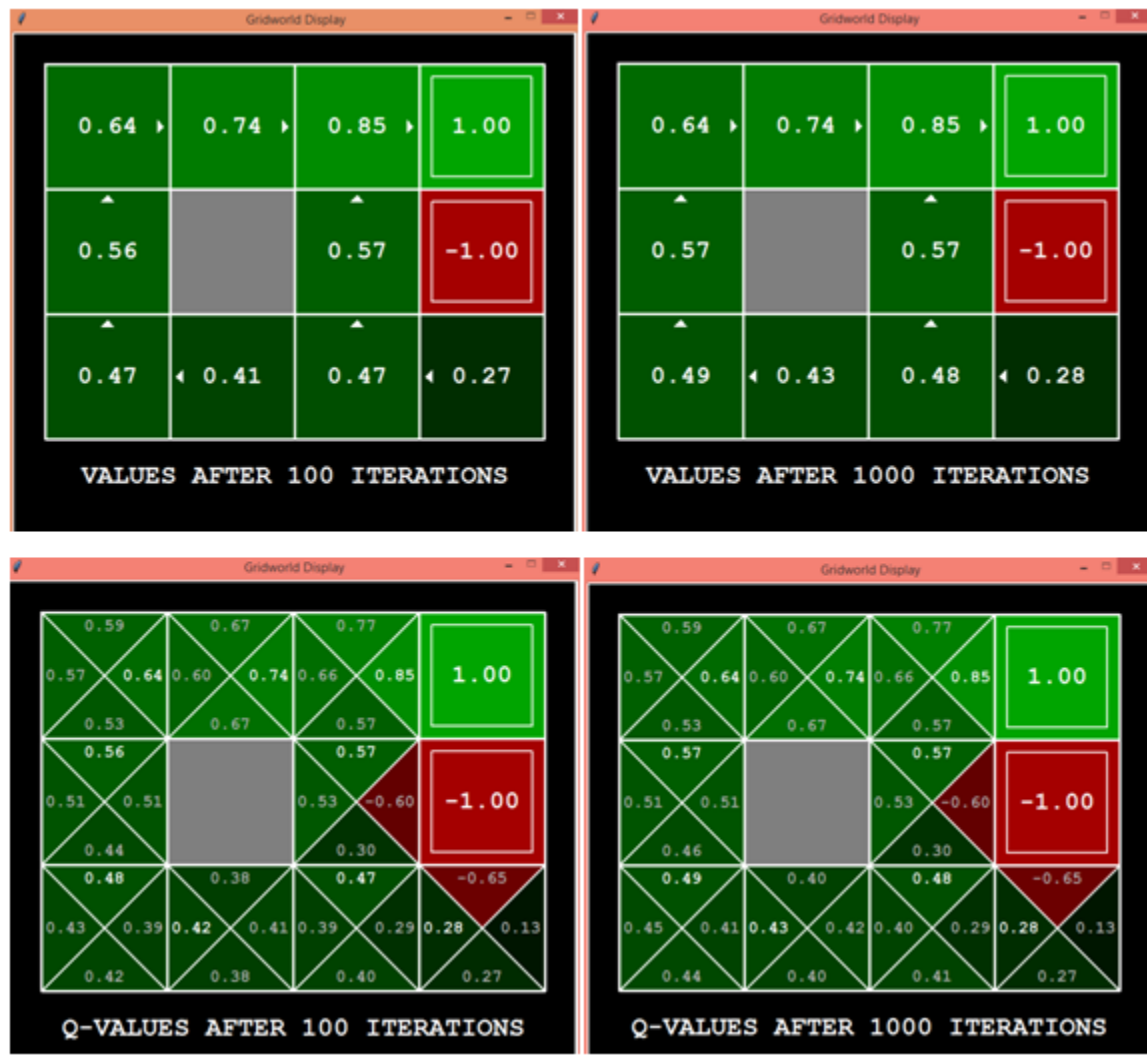
Started in state: (2, 2)
Took action: east
Ended in state: (3, 2)
Got reward: 0.0

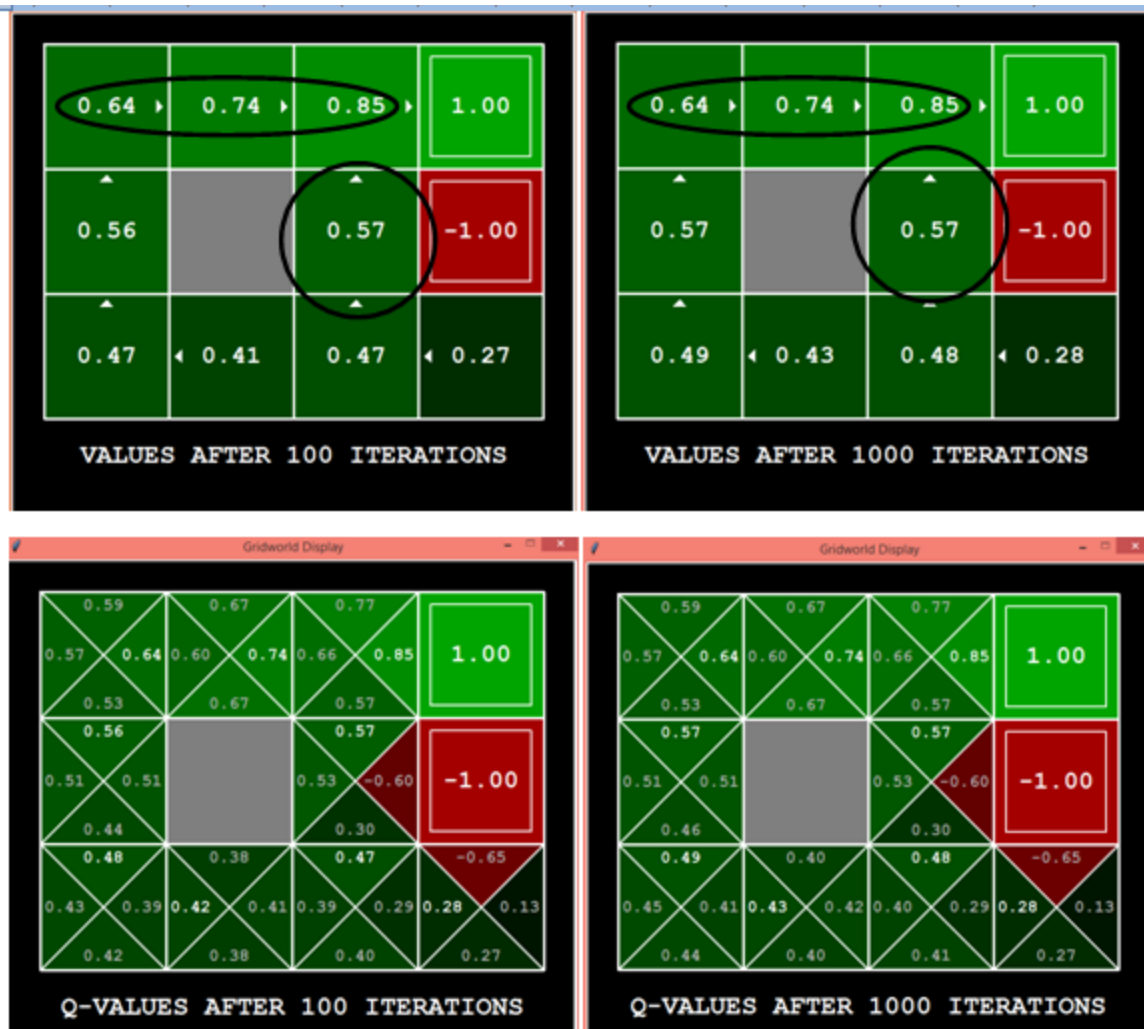
Started in state: (3, 2)
Took action: exit
Ended in state: TERMINAL_STATE
Got reward: 1

EPISODE 1 COMPLETE: RETURN WAS 0.38742048900000015

AVERAGE RETURNS FROM START STATE: 0.38742048900000015
```

The figure below is the comparison of 100 and 1000 iteration command:



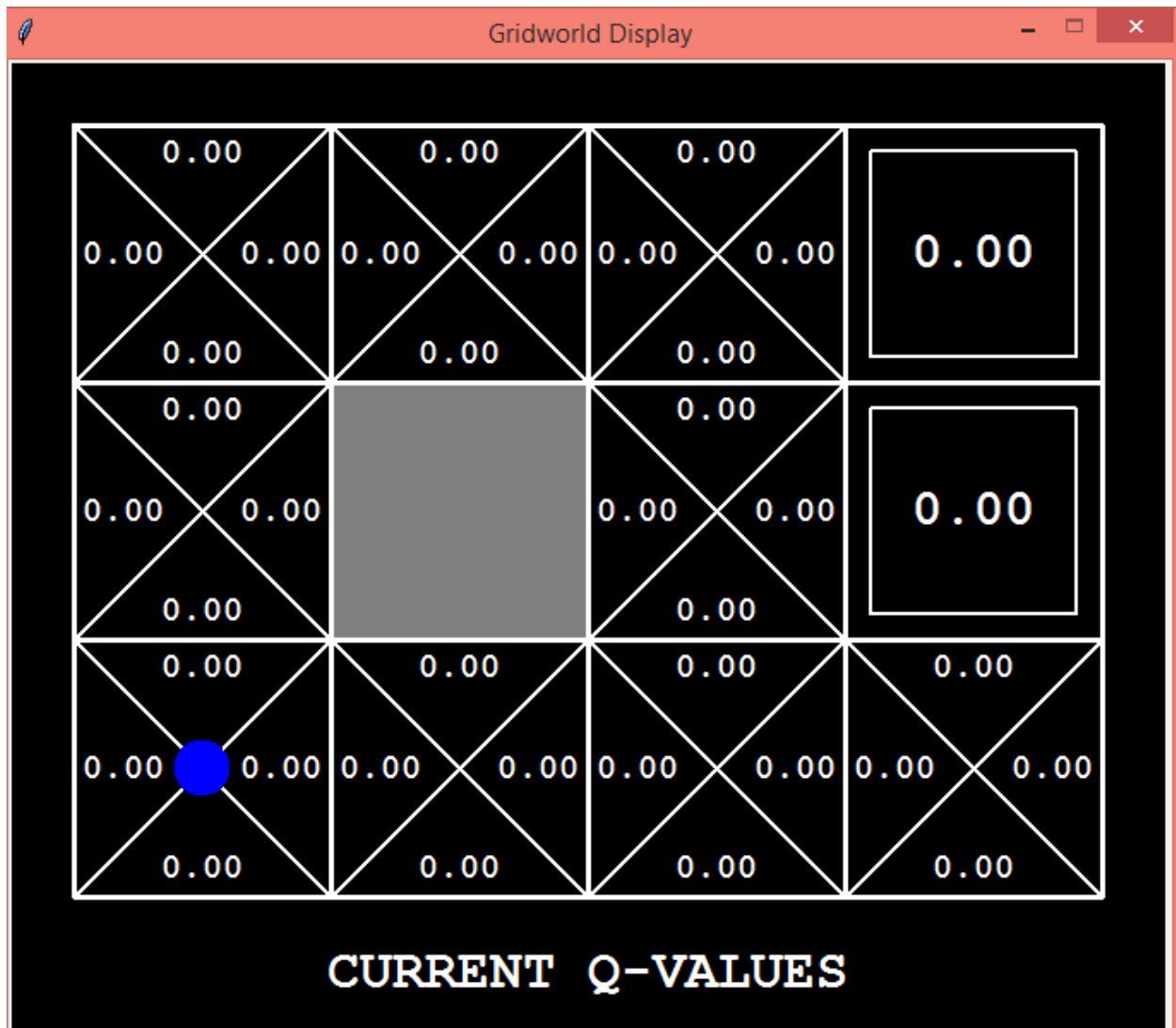


As we can see from the figure some of action state values and Q values are same for both the no. of iterations. But the returning value of both of them differs a bit.

4. Part 2 – Q6 and Q7 - Run Qlearning agent for 100 iterations on default BookGrid and compare the Q-values to those obtained when running value iteration. Run Q-learning with a few variations of noise, epsilon, and discount parameters. Show screenshots and explain the behavior/influence of parameters.

## Prioritized Sweeping Value Iteration

```
python gridworld.py -a q -k 5 -m
```



```
Command Prompt

Started in state: (2, 2)
Took action: east
Ended in state: (2, 1)
Got reward: 0.0

Started in state: (2, 1)
Took action: east
Ended in state: (3, 1)
Got reward: 0.0

Started in state: (3, 1)
Took action: exit
Ended in state: TERMINAL_STATE
Got reward: -1

EPISODE 4 COMPLETE: RETURN WAS -0.5314410000000002

BEGINNING EPISODE: 5

Started in state: (0, 0)
Took action: east
Ended in state: (1, 0)
Got reward: 0.0

Started in state: (1, 0)
Took action: east
Ended in state: (2, 0)
Got reward: 0.0

Started in state: (2, 0)
Took action: north
Ended in state: (2, 1)
Got reward: 0.0

Started in state: (2, 1)
Took action: north
Ended in state: (2, 2)
Got reward: 0.0

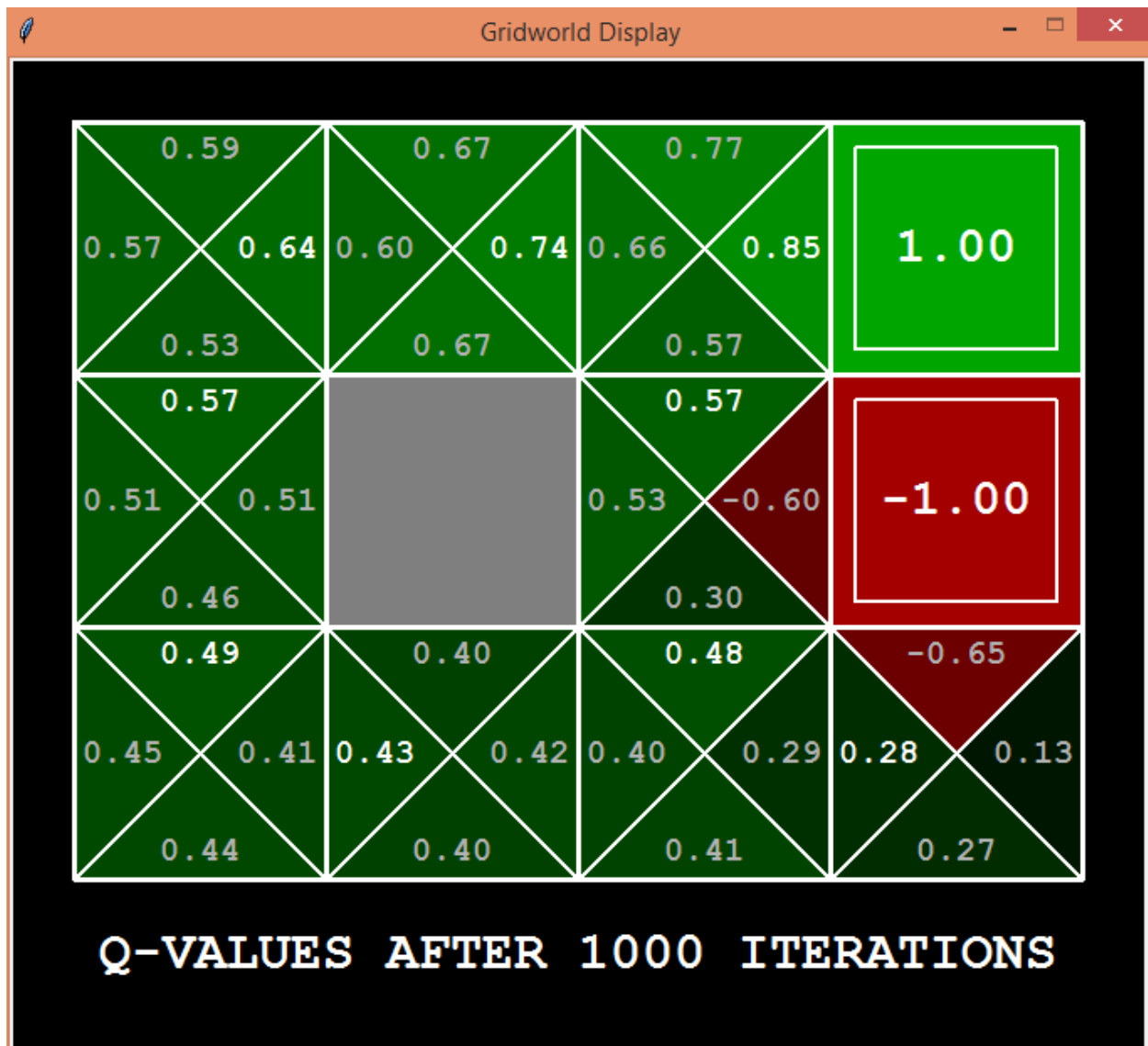
Started in state: (2, 2)
Took action: east
Ended in state: (3, 2)
Got reward: 0.0

Started in state: (3, 2)
Took action: exit
Ended in state: TERMINAL_STATE
Got reward: 1

EPISODE 5 COMPLETE: RETURN WAS 0.5904900000000002

AVERAGE RETURNS FROM START STATE: 0.2702432872962001
```







```
Command Prompt

Started in state: <1, 2>
Took action: east
Ended in state: <2, 2>
Got reward: 0.0

Started in state: <2, 2>
Took action: east
Ended in state: <3, 2>
Got reward: 0.0

Started in state: <3, 2>
Took action: exit
Ended in state: TERMINAL_STATE
Got reward: 1

EPISODE 9 COMPLETE: RETURN WAS 0.5904900000000002

BEGINNING EPISODE: 10

Started in state: <0, 0>
Took action: north
Ended in state: <0, 1>
Got reward: 0.0

Started in state: <0, 1>
Took action: north
Ended in state: <0, 2>
Got reward: 0.0

Started in state: <0, 2>
Took action: east
Ended in state: <1, 2>
Got reward: 0.0

Started in state: <1, 2>
Took action: east
Ended in state: <2, 2>
Got reward: 0.0

Started in state: <2, 2>
Took action: east
Ended in state: <3, 2>
Got reward: 0.0

Started in state: <3, 2>
Took action: exit
Ended in state: TERMINAL_STATE
Got reward: 1

EPISODE 10 COMPLETE: RETURN WAS 0.5904900000000002

AVERAGE RETURNS FROM START STATE: 0.5069705039100001
```

```
python autograder.py -q q6
```

```
C:\Users\HP\Desktop\PUtry2>python autograder.py -q q6
C:\Users\HP\Desktop\PUtry2>autograder.py:17: DeprecationWarning: the imp
is deprecated in favour of importlib; see the module's documentation for
tive uses
  import imp
Starting on 3-7 at 15:06:22

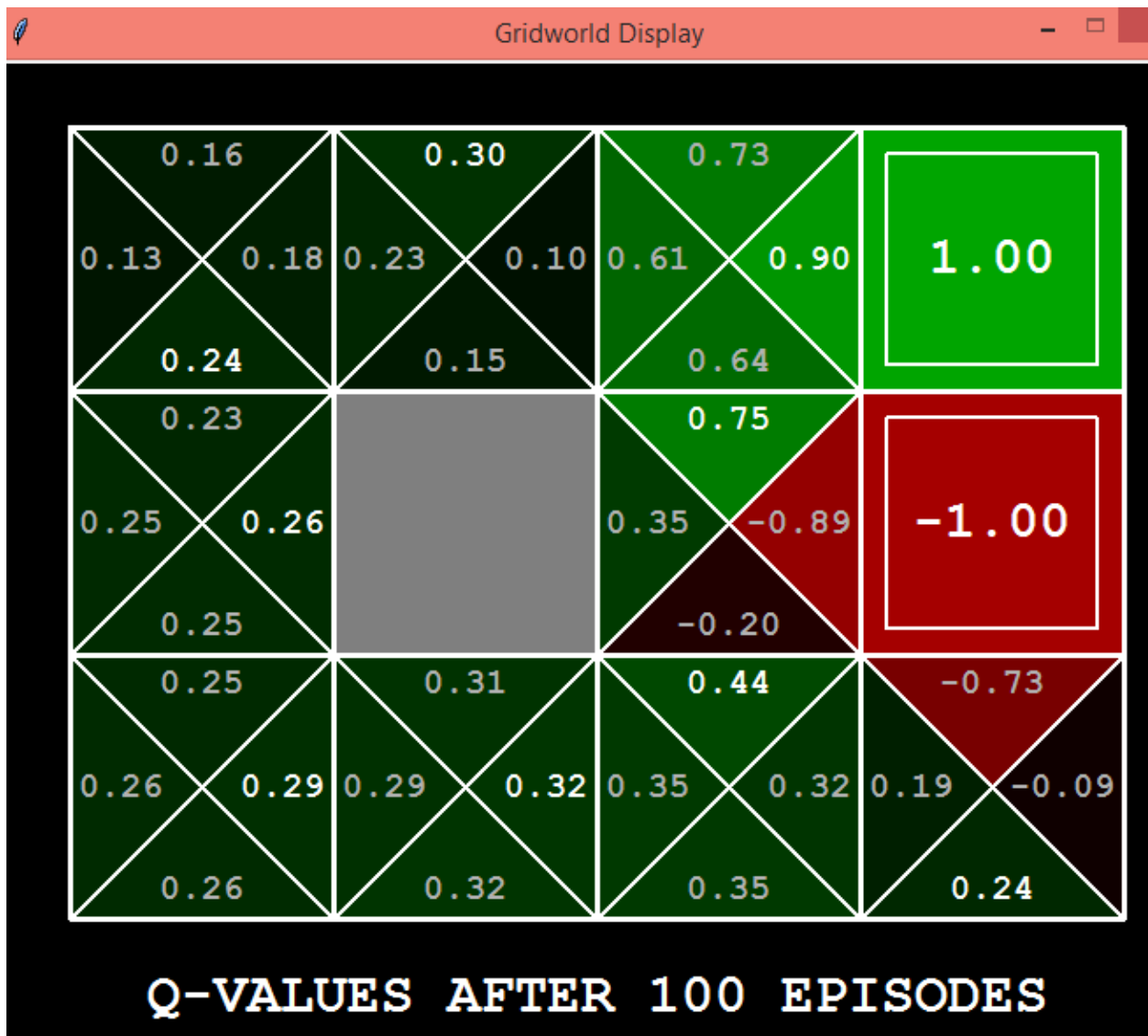
Question q6
=====

*** PASS: test_cases\q6\1-tinygrid.test
```

Implementation of Q.6 is successful.

## Q-Learning

```
python gridworld.py -a q -k 100
```





```
Command Prompt
C:\
Started in state: (0, 0)
Took action: south
Ended in state: (0, 0)
Got reward: 0.0

Started in state: (0, 0)
Took action: east
Ended in state: (1, 0)
Got reward: 0.0

Started in state: (1, 0)
Took action: south
Ended in state: (1, 0)
Got reward: 0.0

Started in state: (1, 0)
Took action: east
Ended in state: (2, 0)
Got reward: 0.0

Started in state: (2, 0)
Took action: north
Ended in state: (2, 1)
Got reward: 0.0

Started in state: (2, 1)
Took action: north
Ended in state: (2, 2)
Got reward: 0.0

Started in state: (2, 2)
Took action: south
Ended in state: (2, 1)
Got reward: 0.0

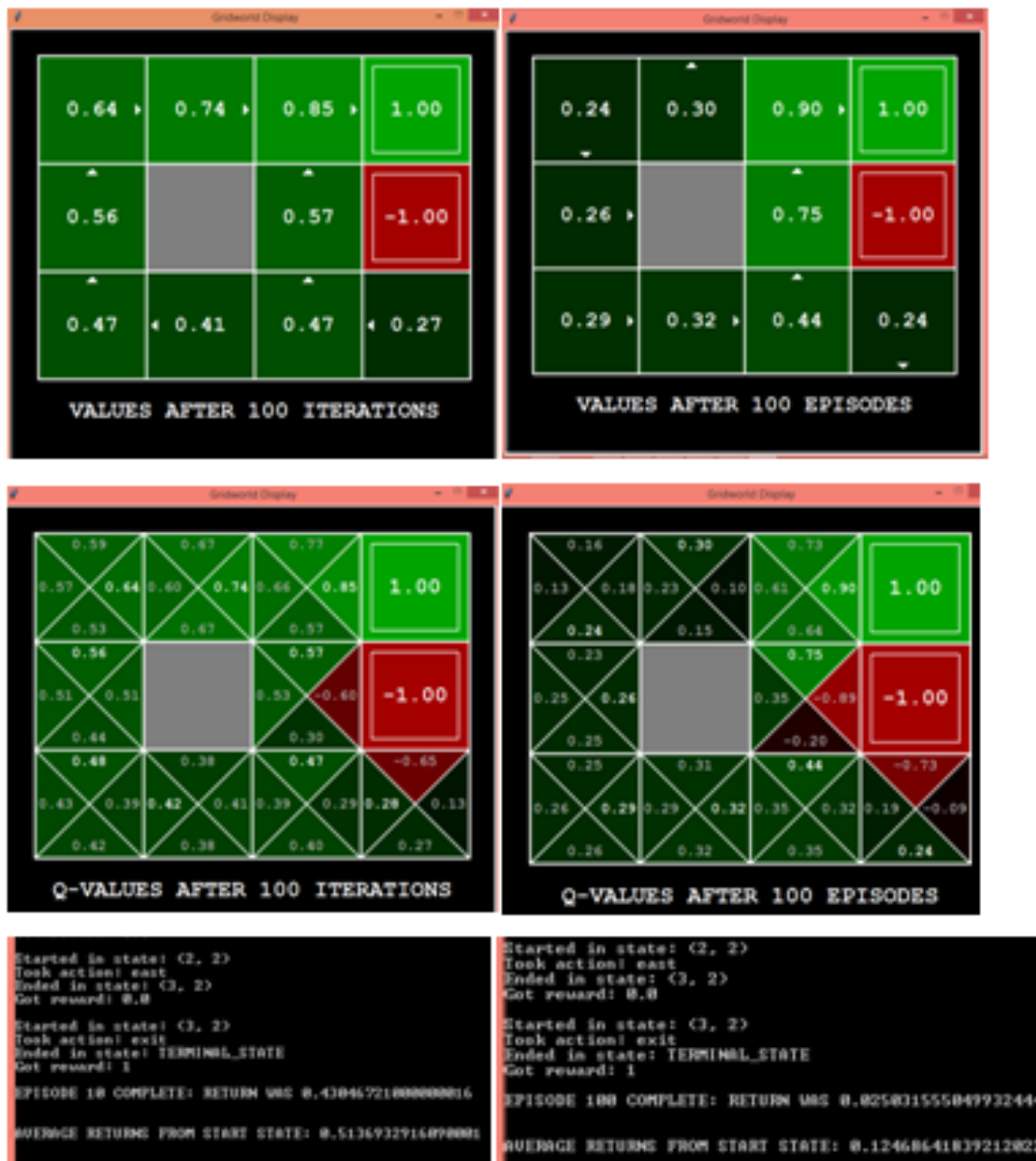
Started in state: (2, 1)
Took action: north
Ended in state: (2, 2)
Got reward: 0.0

Started in state: (2, 2)
Took action: east
Ended in state: (3, 2)
Got reward: 0.0

Started in state: (3, 2)
Took action: exit
Ended in state: TERMINAL_STATE
Got reward: 1

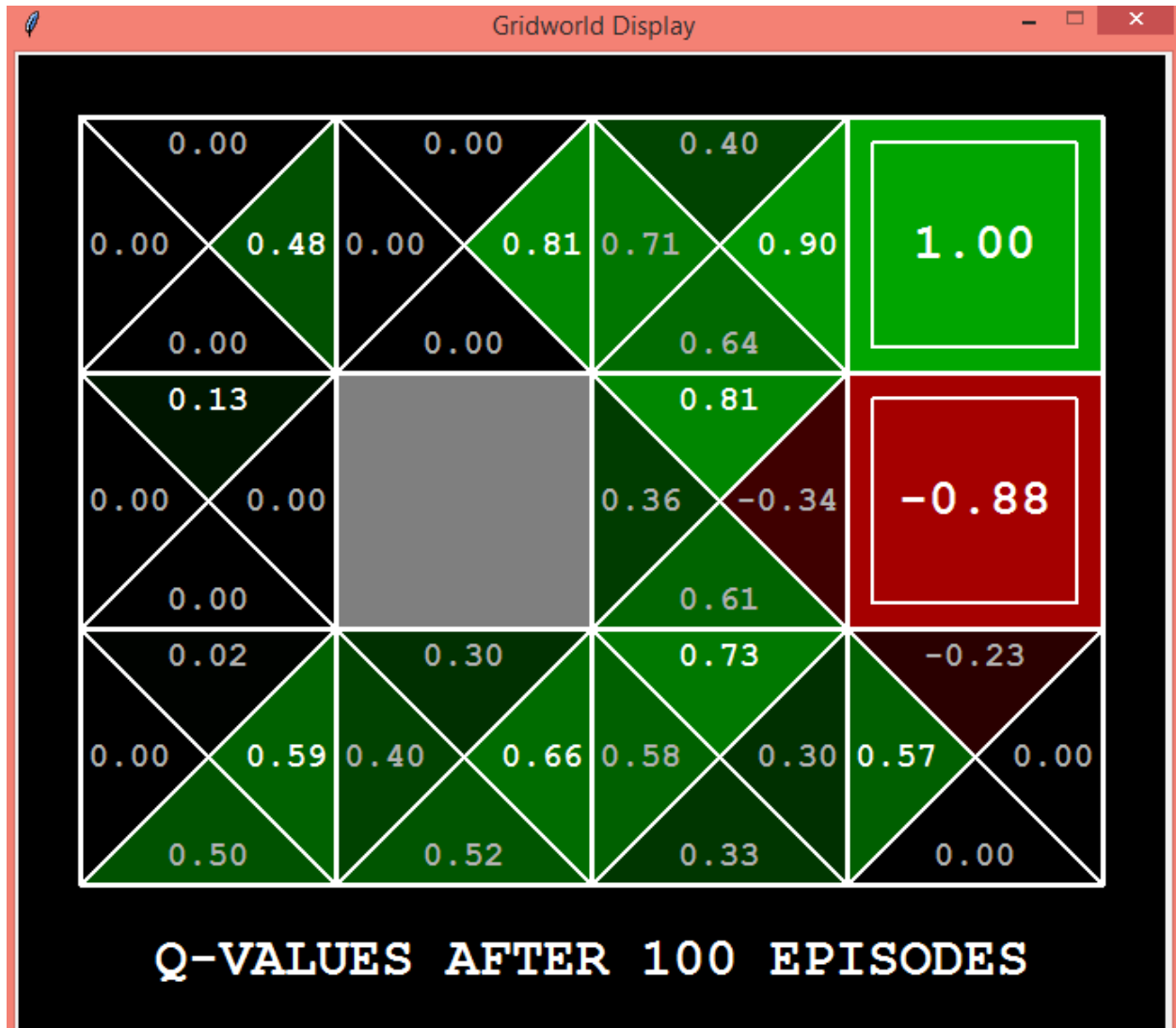
EPISODE 100 COMPLETE: RETURN WAS 0.025031555049932444

AVERAGE RETURNS FROM START STATE: 0.12468641839212022
```



In the above figure is the comparison of the maze grid and Q learning result. The result totally varies because the reverse procedure is used while Q learning. But Q learning results are the most efficient ones.

```
python gridworld.py -a q -k 100 --noise 0.0 -e 0.1
```







```
Command Prompt

Started in state: (2, 1)
Took action: north
Ended in state: (2, 2)
Got reward: 0.0

Started in state: (2, 2)
Took action: east
Ended in state: (3, 2)
Got reward: 0.0

Started in state: (3, 2)
Took action: exit
Ended in state: TERMINAL_STATE
Got reward: 1

EPISODE 99 COMPLETE: RETURN WAS 0.5904900000000002

BEGINNING EPISODE: 100

Started in state: (0, 0)
Took action: east
Ended in state: (1, 0)
Got reward: 0.0

Started in state: (1, 0)
Took action: east
Ended in state: (2, 0)
Got reward: 0.0

Started in state: (2, 0)
Took action: north
Ended in state: (2, 1)
Got reward: 0.0

Started in state: (2, 1)
Took action: north
Ended in state: (2, 2)
Got reward: 0.0

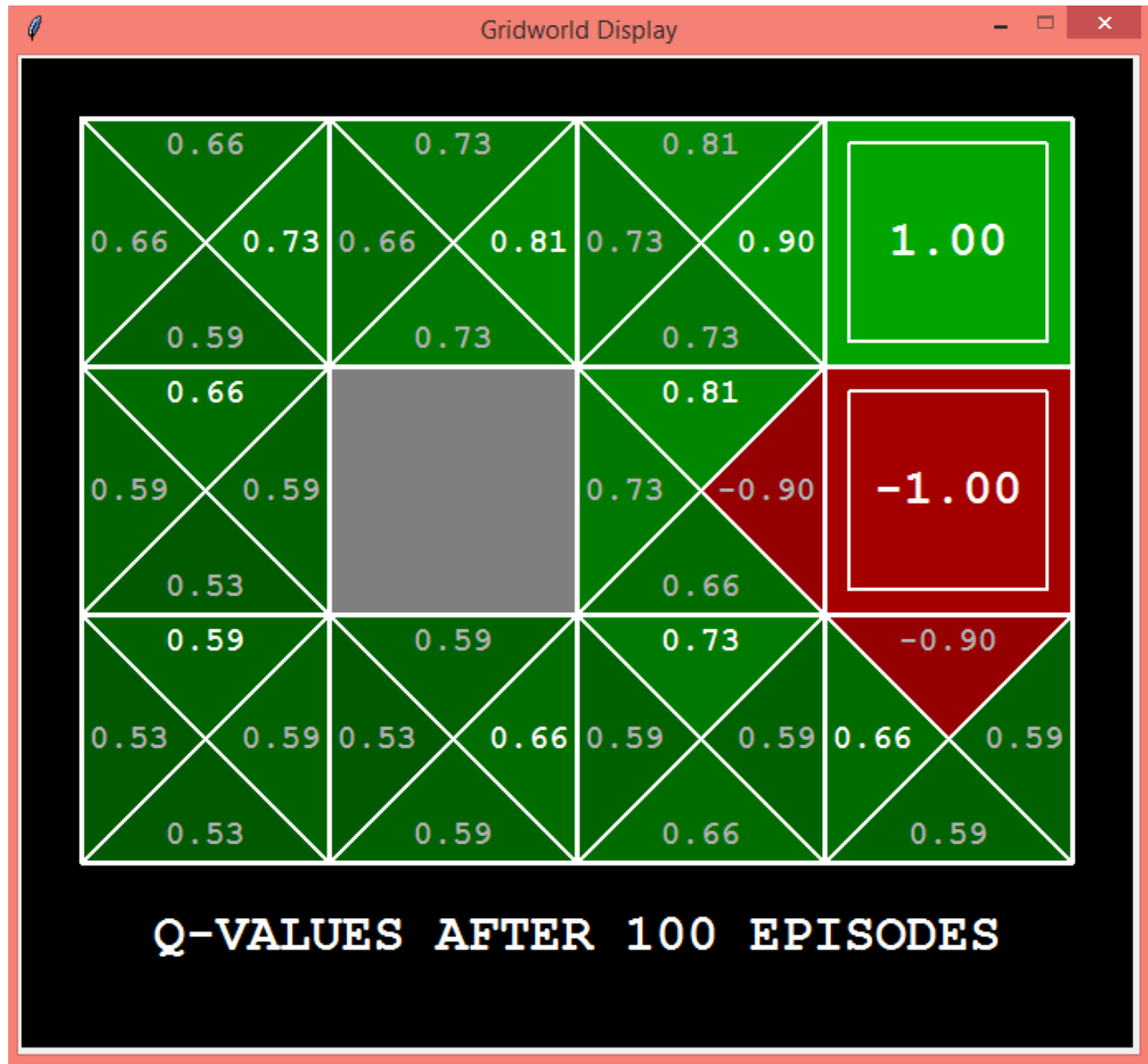
Started in state: (2, 2)
Took action: east
Ended in state: (3, 2)
Got reward: 0.0

Started in state: (3, 2)
Took action: exit
Ended in state: TERMINAL_STATE
Got reward: 1

EPISODE 100 COMPLETE: RETURN WAS 0.5904900000000002

AVERAGE RETURNS FROM START STATE: 0.4985986973184696
```

```
python gridworld.py -a q -k 100 --noise 0.0 -e 0.9
```





```
C:\ Command Prompt

Started in state: <3, 2>
Took action: exit
Ended in state: TERMINAL_STATE
Got reward: 1

EPISODE 99 COMPLETE: RETURN WAS 0.07178979876918531

BEGINNING EPISODE: 100

Started in state: <0, 0>
Took action: south
Ended in state: <0, 0>
Got reward: 0.0

Started in state: <0, 0>
Took action: east
Ended in state: <1, 0>
Got reward: 0.0

Started in state: <1, 0>
Took action: west
Ended in state: <0, 0>
Got reward: 0.0

Started in state: <0, 0>
Took action: east
Ended in state: <1, 0>
Got reward: 0.0

Started in state: <1, 0>
Took action: east
Ended in state: <2, 0>
Got reward: 0.0

Started in state: <2, 0>
Took action: east
Ended in state: <3, 0>
Got reward: 0.0

Started in state: <3, 0>
Took action: north
Ended in state: <3, 1>
Got reward: 0.0

Started in state: <3, 1>
Took action: exit
Ended in state: TERMINAL_STATE
Got reward: -1

EPISODE 100 COMPLETE: RETURN WAS -0.47829690000000014

AVERAGE RETURNS FROM START STATE: -0.003948527678820025
```

## Epsilon Greedy

```
python autograder.py -q q7
```

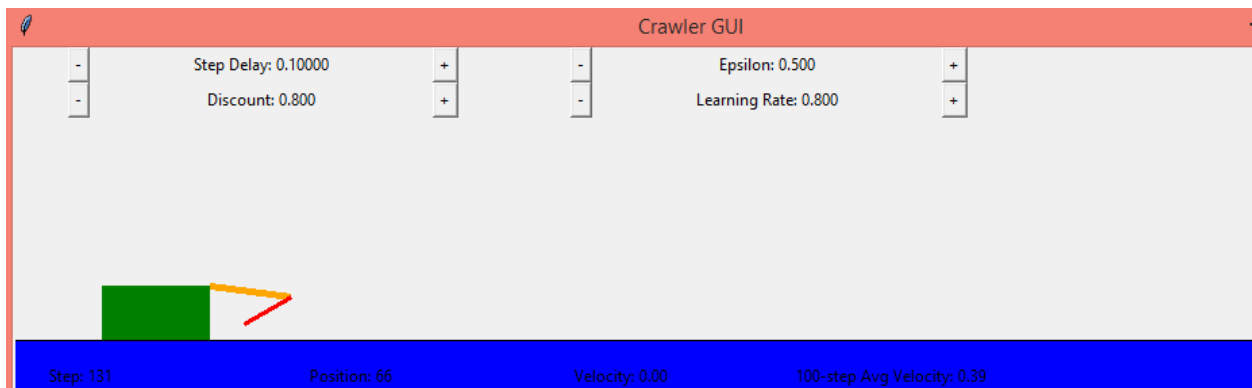
```
C:\Users\HP\Desktop\PUtry2>python autograder.py -q q7
C:\Users\HP\Desktop\PUtry2>autograder.py:17: DeprecationWarning: the
is deprecated in favour of importlib; see the module's documentatio
tive uses
import imp
Starting on 3-7 at 15:24:26

Question q7
=====
*** PASS: test_cases\q7\1-tinygrid-test
```

Working of Q. 7 is successfully implemented.

Also, Tried the code for the crawler game and it was working fine.

```
python crawler.py
```



```
python gridworld.py -a q -k 50 -n 0 -g BridgeGrid -e 1
```



## Bridge Crossing Revisited

```
python autograder.py -q q8
```

```
Question q8
=====
*** PASS: test_cases\q8\grade-agent.test
```

Implementation of Q. 8 is successful.

5. Part 2- Q9 - Run Q-learning in Pacman, as specified in Q9 from the Pacman RL project, using the parameters specified in the question (2000 iterations, small grid). Show screenshots of your results (average reward, and win/loses). Now run it for 2000 iterations in the mediumGrid in Pacman and show the results. What do you observe? Increase the number of training iterations (and if needed modify epsilon) to achieve desired Pacman performance in medium grid. Show screenshots of results and list which parameters/number of iterations did you have to use.

```
python pacman.py -p PacmanQAgent -x 2000 -n 2010 -l smallGrid
```



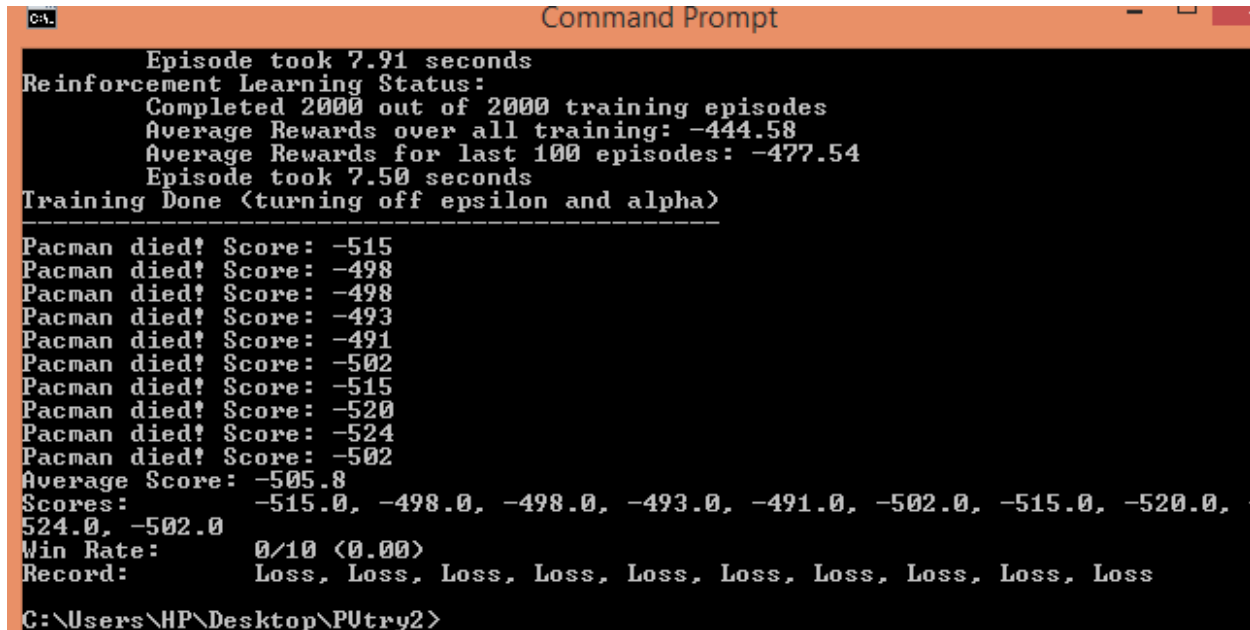
```

Command Prompt

Completed 1300 out of 2000 training episodes
Average Rewards over all training: -200.27
Average Rewards for last 100 episodes: 226.84
Episode took 5.43 seconds
Reinforcement Learning Status:
Completed 1400 out of 2000 training episodes
Average Rewards over all training: -177.15
Average Rewards for last 100 episodes: 123.42
Episode took 7.40 seconds
Reinforcement Learning Status:
Completed 1500 out of 2000 training episodes
Average Rewards over all training: -150.28
Average Rewards for last 100 episodes: 225.92
Episode took 7.30 seconds
Reinforcement Learning Status:
Completed 1600 out of 2000 training episodes
Average Rewards over all training: -127.41
Average Rewards for last 100 episodes: 215.56
Episode took 5.65 seconds
Reinforcement Learning Status:
Completed 1700 out of 2000 training episodes
Average Rewards over all training: -103.03
Average Rewards for last 100 episodes: 287.06
Episode took 6.44 seconds
Reinforcement Learning Status:
Completed 1800 out of 2000 training episodes
Average Rewards over all training: -85.88
Average Rewards for last 100 episodes: 205.80
Episode took 6.75 seconds
Reinforcement Learning Status:
Completed 1900 out of 2000 training episodes
Average Rewards over all training: -68.92
Average Rewards for last 100 episodes: 236.29
Episode took 7.81 seconds
Reinforcement Learning Status:
Completed 2000 out of 2000 training episodes
Average Rewards over all training: -57.67
Average Rewards for last 100 episodes: 156.01
Episode took 3.67 seconds
Training Done (turning off epsilon and alpha)
-----
Pacman emerges victorious! Score: 495
Pacman emerges victorious! Score: 501
Pacman emerges victorious! Score: 501
Pacman emerges victorious! Score: 501
Pacman emerges victorious! Score: 501
Pacman emerges victorious! Score: 499
Pacman emerges victorious! Score: 503
Pacman emerges victorious! Score: 495
Pacman emerges victorious! Score: 503
Pacman emerges victorious! Score: 499
Average Score: 499.8
Scores: 495.0, 501.0, 501.0, 501.0, 501.0, 499.0, 503.0, 495.0, 503.0, 499.0
Win Rate: 10/10 (1.00)
Record: Win, Win, Win, Win, Win, Win, Win, Win, Win, Win
C:\Users\HP\Desktop>PHTpu2>

```

```
python pacman.py -p PacmanQAgent -x 2000 -n 2010 -l mediumGrid
```



```
Episode took 7.91 seconds
Reinforcement Learning Status:
  Completed 2000 out of 2000 training episodes
  Average Rewards over all training: -444.58
  Average Rewards for last 100 episodes: -477.54
  Episode took 7.50 seconds
Training Done (turning off epsilon and alpha)
-----
Pacman died! Score: -515
Pacman died! Score: -498
Pacman died! Score: -498
Pacman died! Score: -493
Pacman died! Score: -491
Pacman died! Score: -502
Pacman died! Score: -515
Pacman died! Score: -520
Pacman died! Score: -524
Pacman died! Score: -502
Average Score: -505.8
Scores:      -515.0, -498.0, -498.0, -493.0, -491.0, -502.0, -515.0, -520.0, -
524.0, -502.0
Win Rate:    0/10 (0.00)
Record:      Loss, Loss, Loss, Loss, Loss, Loss, Loss, Loss, Loss, Loss
C:\Users\HP\Desktop\PUtry2>
```

With 2000 iterations on medium grid the pacman is dying. We have tried to increase the number of iterations to maximize the training.

But it is observed that, Pacman will begin to learn about the values of positions and actions. Because it takes a very long time to learn accurate Q-values even for tiny grids. So to make it successful with medium grade we will have to provide good number of training iterations.

```
python autograder.py -q q9
```

```

C:\ Command Prompt
Pacman emerges victorious! Score: 503
Pacman emerges victorious! Score: 499
Pacman emerges victorious! Score: 499
Pacman emerges victorious! Score: 503
Pacman emerges victorious! Score: 503
Pacman emerges victorious! Score: 495
Pacman emerges victorious! Score: 499
Pacman emerges victorious! Score: 503
Pacman emerges victorious! Score: 495
Pacman emerges victorious! Score: 503
Pacman emerges victorious! Score: 503
Pacman emerges victorious! Score: 503
Pacman emerges victorious! Score: 495
Pacman emerges victorious! Score: 503
Pacman emerges victorious! Score: 503
Pacman emerges victorious! Score: 495
Pacman emerges victorious! Score: 503
Pacman emerges victorious! Score: 503
Pacman emerges victorious! Score: 495
Pacman emerges victorious! Score: 503
Pacman emerges victorious! Score: 503
Pacman emerges victorious! Score: 495
Pacman emerges victorious! Score: 499
Pacman emerges victorious! Score: 499
Pacman emerges victorious! Score: 503
Pacman emerges victorious! Score: 499
Pacman emerges victorious! Score: 499
Pacman emerges victorious! Score: 495
Pacman emerges victorious! Score: 503
Pacman emerges victorious! Score: 503
Pacman emerges victorious! Score: 495
Pacman emerges victorious! Score: 499
Pacman emerges victorious! Score: 503
Pacman emerges victorious! Score: 503
Reinforcement Learning Status:
    Completed 100 test episodes
    Average Rewards over testing: 499.96
    Average Rewards for last 100 episodes: 499.96
    Episode took 5.46 seconds
Average Score: 499.96
Scores:      503.0, 503.0, 503.0, 503.0, 499.0, 499.0, 495.0, 495.0, 503.0, 49
9.0, 495.0, 495.0, 495.0, 503.0, 495.0, 495.0, 503.0, 495.0, 503.0, 503.0, 503.0
, 503.0, 499.0, 495.0, 495.0, 503.0, 503.0, 499.0, 503.0, 503.0, 503.0, 499.0, 4
99.0, 495.0, 503.0, 499.0, 503.0, 495.0, 499.0, 503.0, 499.0, 503.0, 503.0, 499.
0, 495.0, 503.0, 499.0, 503.0, 499.0, 503.0, 495.0, 503.0, 499.0, 499.0, 495.0,
503.0, 503.0, 499.0, 495.0, 495.0, 503.0, 495.0, 503.0, 503.0, 503.0, 495.0, 503
.0, 499.0, 499.0, 503.0, 503.0, 495.0, 499.0, 503.0, 495.0, 503.0, 503.0, 503.0,
503.0, 495.0, 503.0, 503.0, 495.0, 503.0, 503.0, 503.0, 495.0, 499.0, 499.0, 50
3.0, 499.0, 499.0, 495.0, 503.0, 503.0, 495.0, 499.0, 503.0, 503.0
Win Rate:    100/100 (1.00)
Record:      Win, Win, Win, Win, Win, Win, Win, Win, Win, Win, Win, Win, Win, Win,
Win, Win, Win, Win, Win, Win, Win, Win, Win, Win, Win, Win, Win, Win, Win, Win,
Win, Win, Win, Win, Win, Win, Win, Win, Win, Win, Win, Win, Win, Win, Win, Win,
Win, Win, Win, Win, Win, Win, Win, Win, Win, Win, Win, Win, Win, Win, Win, Win,
Win, Win, Win, Win, Win, Win, Win, Win, Win, Win, Win, Win, Win, Win, Win, Win,
Win, Win, Win, Win, Win, Win, Win
*** PASS: test_cases\q9\grade-agent.test (1 of 1 points)
```

The above output proves that Q.9 is also successfully implemented.