Homework 8

Start Assignment

Due Apr 26 by 5pm **Points** 18 **Submitting** a file upload **File Types** py
Available until Apr 28 at 5pm

Your task this week is to guide our agent to a treasure hidden at the bottom of the ocean.

The task is presented as a game and the mechanics have been implemented for you. The starter code is available in **HW8.zip** (https://sjsu.instructure.com/courses/1420011/files/63047836/download?download_frd=1) .

Once you download the code, you can play the game in discovery mode (without help) by typing:

python treasurehunt.py 6 discovery

To play the game on a bigger grid, you can try:

python treasurehunt.py 10 discovery

Click on the 'SONAR' button to start taking sonar measurements. When you are ready to make a decision and dive, click on the 'DIVE' button.

Your task is to help the player/agent find the treasure more reliably by implementing the following:

1. The *update* method in the beliefs.py module. The Belief class is used to track the belief distribution (where we think the treasure is) based on the sonar sensing evidence we have so far. The distribution is initialized to a uniform distribution in __init__. Your task is to update it with each evidence. Don't forget to normalize!

   To see the distribution on the grid, start the game in 'guided' (default) mode:

   python treasurehunt.py 8 guided

   Note that the player should be able to sense the same location more than once.

2. The r*ecommend_sensing* method in the beliefs.py module. The method returns the position where we should take the next sonar measurement in the grid. The position should be the most promising unobserved location. If all remaining unobserved locations have a probability of 0, the

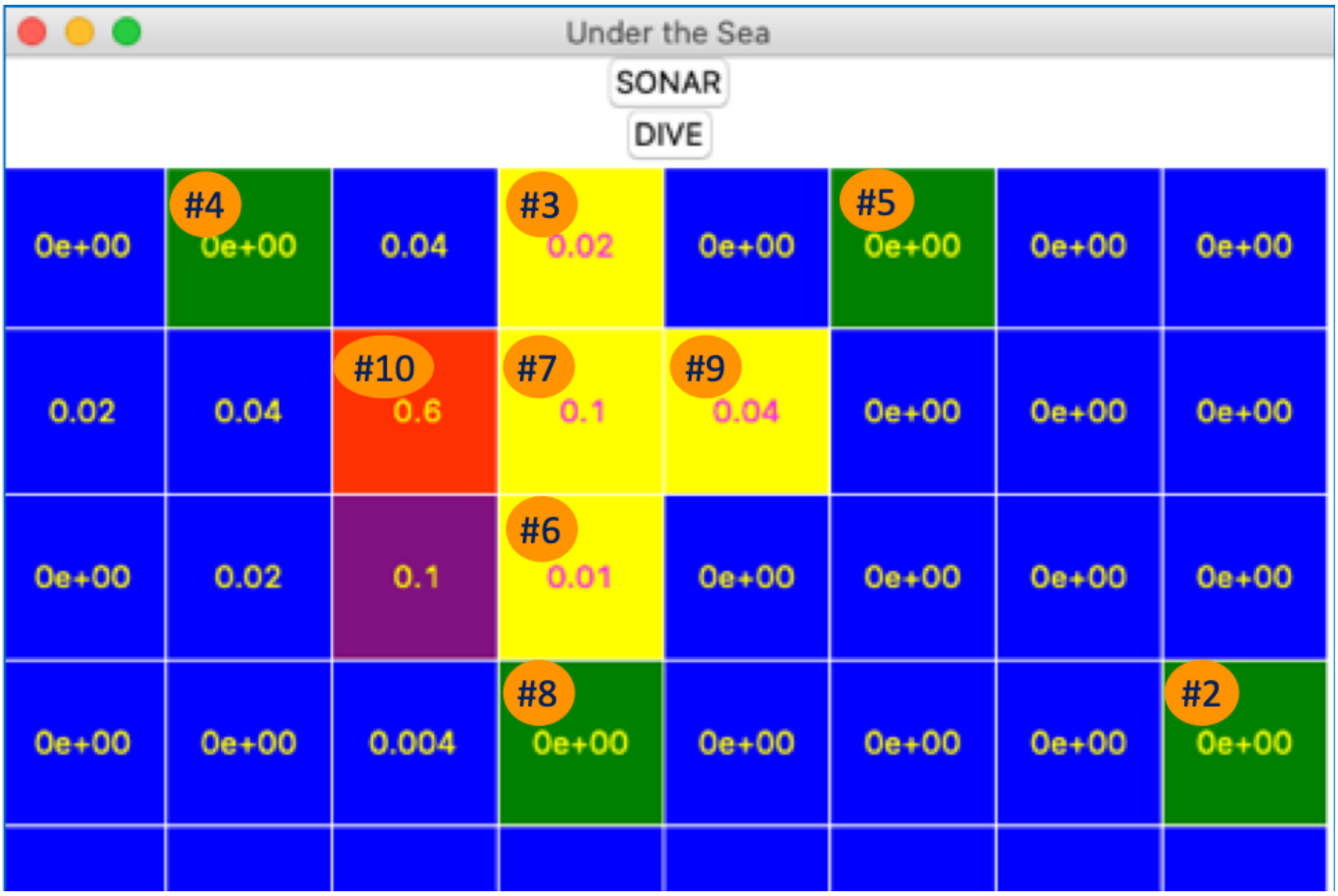
method should return the unobserved location that is closest to the (observed) location with the highest probability. The recommended location is displayed in purple on the grid - as shown in the screenshots below. If there are no remaining unobserved locations, the method will return the (observed) location with the highest probability. Note that you may not be able to 'see' the sonar color after sensing if the same location is recommended next. That is ok because the goal here is to figure out where the treasure is without sensing everywhere.

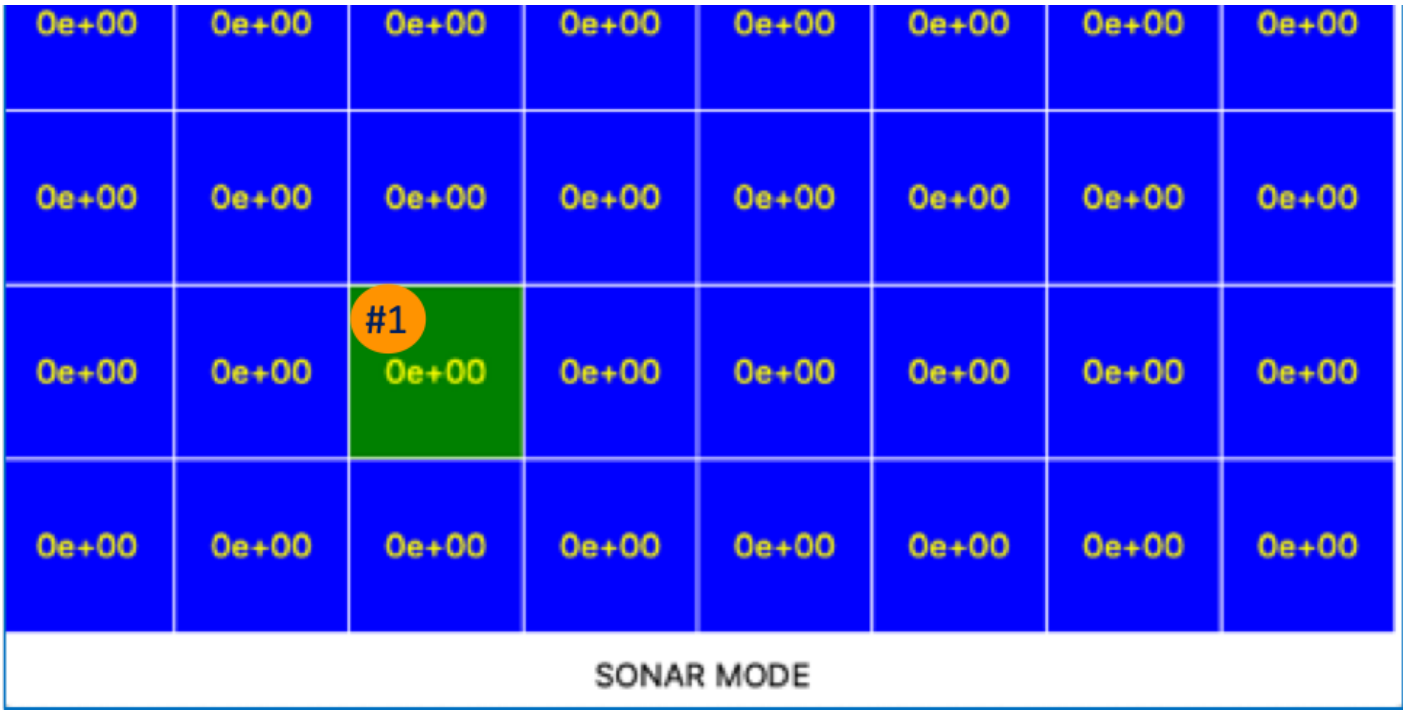
Testing

To test your implementation you may do the following:

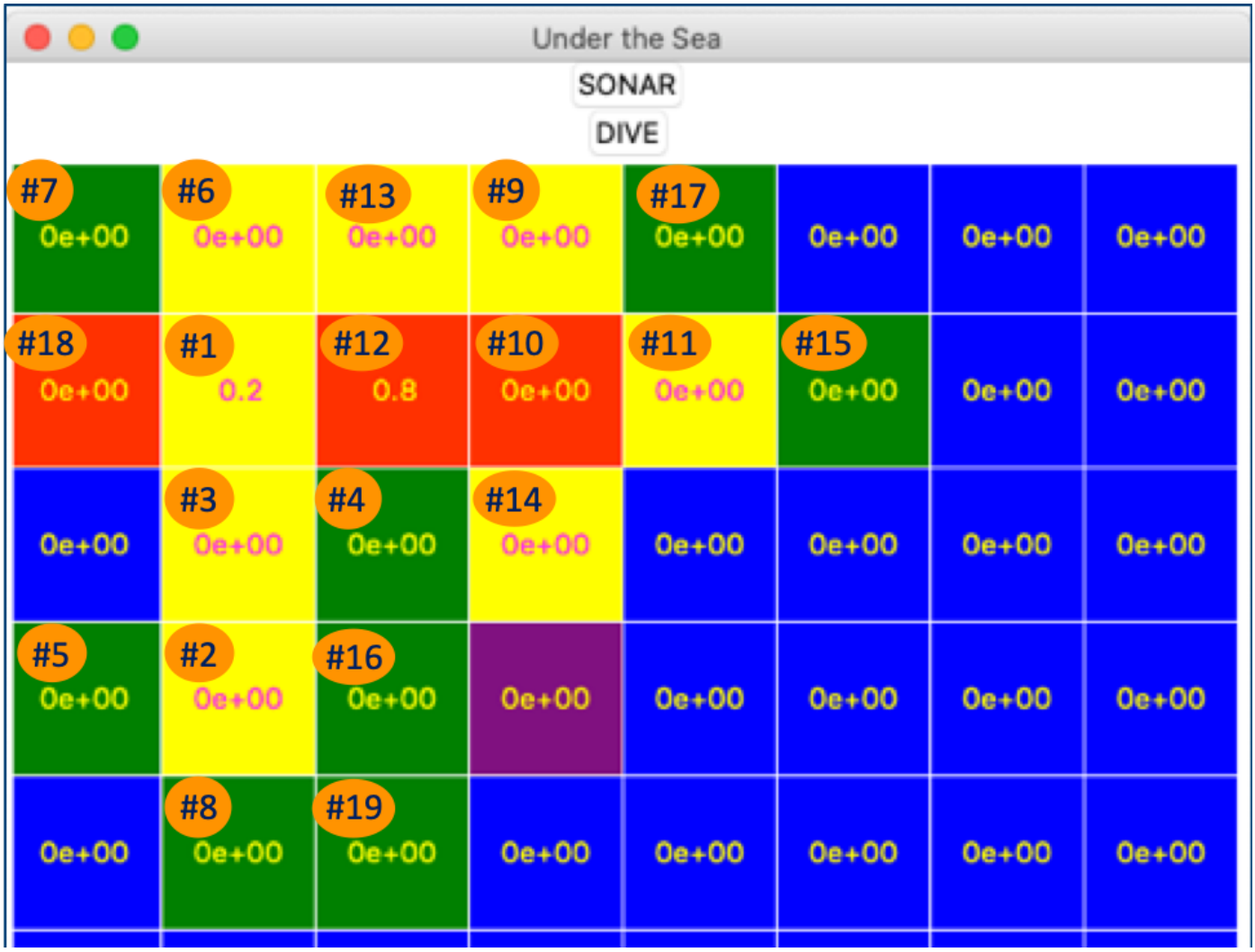
1. Modify the *main* function in the file treasurehunt.py to remove the randomness: uncomment the statement: random.seed(1)
2. Start the program and select the positions shown in the 3 scenarios shown below - in order - in sonar mode. The distribution you get after sensing the numbered positions shown should match. Note that the game will be boring when you do that - since the treasure will always be in the same location.

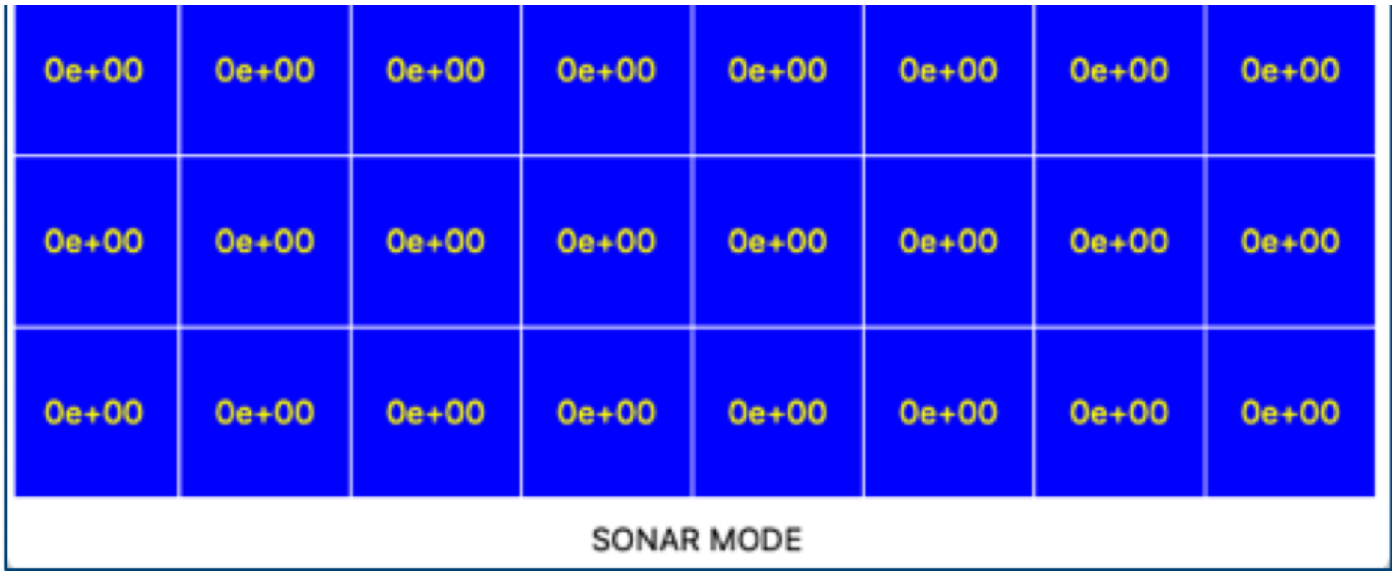
Scenario 1: python treasurehunt.py 8 guided



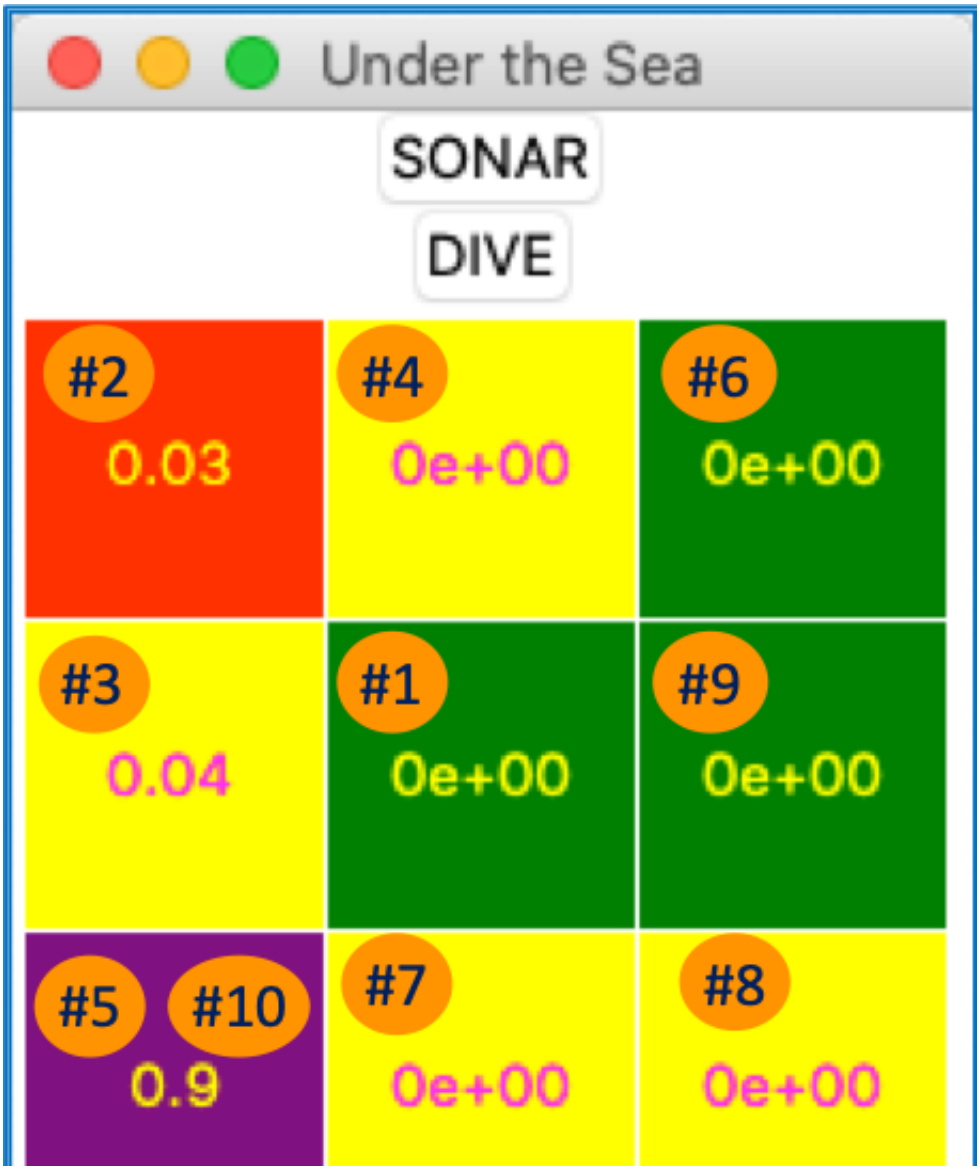


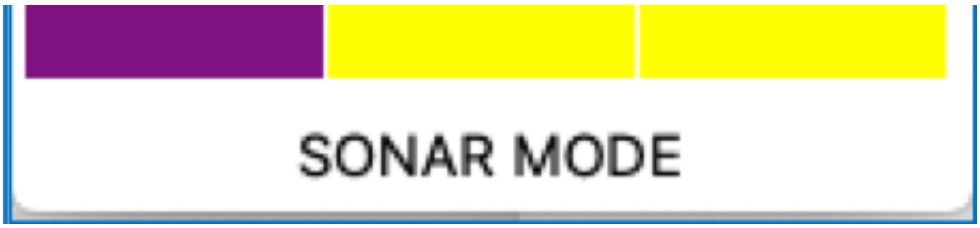
Scenario 2: python treasurehunt.py 8 guided





Scenario 3 with a small grid: python treasurehunt.py 3 guided





You only need to modify and upload beliefs.py.

Please make sure you read the grading rubric to ensure full credit.

Homework 8 Grading Rubric			
Criteria	**Ratings**		**Pts**
The update method: distribution is updated correctly based on the model	**3 pts Full Marks**	**0 pts No Description**	3 pts
The update method: normalization The probabilities for all locations in the grid must add up to 1.	**2 pts Full Marks**	**0 pts No Description**	2 pts
The update method: the open set (unobserved locations) is updated correctly	**2 pts Full Marks**	**0 pts No Description**	2 pts
The update method: the update is correct when the player senses an observed location	**1 pts Full Marks**	**0 pts No Description**	1 pts
The recommend_sensing method: most promising unobserved location An open (unobserved) location with the highest probability is returned	**3 pts Full Marks**	**0 pts No Description**	3 pts
The recommend_sensing method: if all remaining unobserved locations have	**3 pts**	**0 pts**	

a probability of 0, return the unobserved location that is closest to the (observed) location with the highest probability.	**Full Marks**	**No Description**	3 pts
The recommend_sensing method:If there are no remaining unobserved locations, the method will return the (observed) location with the highest probability.	**2 pts Full Marks**	**0 pts No Description**	2 pts
The manhattan_distance function in utils.py is used	**1 pts Full Marks**	**0 pts No Marks**	1 pts
The closest_point function in utils.py is used	**1 pts Full Marks**	**0 pts No Marks**	1 pts
		Total Points: 18	