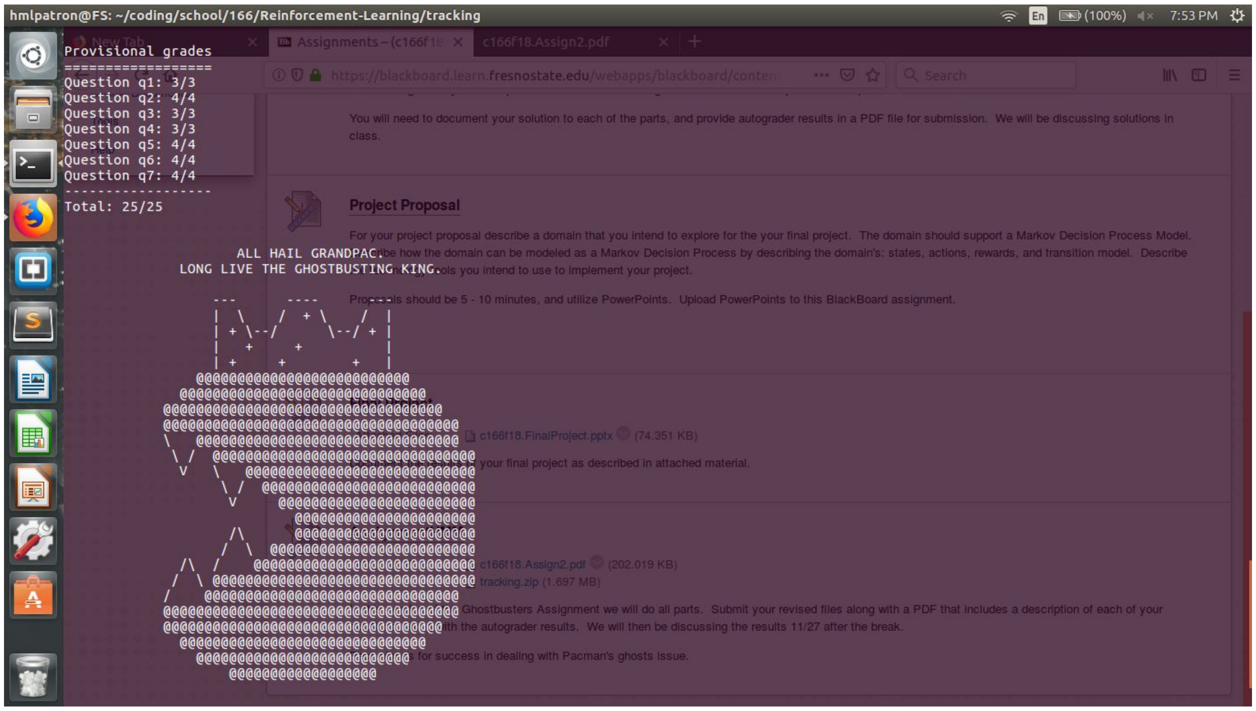
Autograder Score



```
hmlpatron@FS: ~/coding/school/166/Reinforcement-Learning/tracking

Provisional grades
==================
Question q1: 3/3
Question q2: 4/4
Question q3: 3/3
Question q4: 3/3
Question q5: 4/4
Question q6: 4/4
Question q7: 4/4
------------------
Total: 25/25

                     ALL HAIL GRANDPAC.
              LONG LIVE THE GHOSTBUSTING KING.

                  ---      ----
                  |  \    /  + \    /  |
                  | + \--/      \--/ + |
                  |   +     +          |
                  | +    +        +    |
                @@@@@@@@@@@@@@@@@@@@@@@@@@
              @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
            @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
            @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
            \   @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
             \ /  @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
              V   \   @@@@@@@@@@@@@@@@@@@@@@@@@@@@
                   \ /  @@@@@@@@@@@@@@@@@@@@@@@@@@
                    V     @@@@@@@@@@@@@@@@@@@@@@@@
                            @@@@@@@@@@@@@@@@@@@@@@
                    /\      @@@@@@@@@@@@@@@@@@@@@@
                   /  \  @@@@@@@@@@@@@@@@@@@@@@@@@
              /\  /    @@@@@@@@@@@@@@@@@@@@@@@@@@@
             /  \ @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
            /    @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
            @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
            @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
              @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                @@@@@@@@@@@@@@@@@@@@@@@@@@
                    @@@@@@@@@@@@@@@@@@
```

1)
For the first question we were supposed to implement the observe method for the exact inference class. The observe method was responsible for correctly updating the agents belief distribution over the ghost positions given input from a sensor. To do this I first initialized a dictionary container that I called allPossible to represent the current belief as to where the ghost was. I then checked for the edge case. If the ghost was in jail then we knew for sure where he was so we could just assign the dictionary value (probability) for the jail cell to be 1.0 and every other cell to be 0. If the ghost wasn't in jail then we would loop through every position that the ghost could be in and assign a value to that position via the allPossible map. The value of the position was determined by multiplying the current beliefs for that position by the the emission model value at the Manhattan distance to the to the state.

2)
For the second question we implemented the elapsetime method for the ExactInference class. The basic idea here was that Pacman should be able to know things about the way that ghosts are allowed to move (i.e. ghost can't move through walls, can only move 1 time step in a space) in order to make a better estimate at where they might be. The first thing that I did to implement this was to initialize an allPossible dictionary just like the last question. Then I looped through every legal position and for every legal position I got a position distribution using the getPositionDistribution method. From here, I took the tip given to us in the code comments about how to loop over the position/probability dictionary that the last function call gave us. From here I followed Dr. Ruby's advice from office hours about summing out one of the variables so for every position in the new distribution I multiplied the probability for that position

with the old belief value and summed it with the new belief value. This gave me a dictionary of position-beliefs which I then used to overwrite the old beliefs.

3)
The idea for number 3 was to use the methods implemented in the last two questions together in bustersAgents.py. After my initializations, the first thing that I did was loop through all of the living ghost position distributions and take the max value from each one and store the corresponding position. From these possible positions I then found the position that was closest to pacman and sent him towards it.

4)
Number four and onwards were much more difficult for me because for these questions we were implementing multiple components as once so if there was a bug it was more difficult to find where the problem was originating from. The first method that I implemented for number four was the observe function. The first thing that I did was check if the ghost was already in jail. If the ghost was already in jail then I set the value for each particle to be the jail position. If the ghost was still roaming around then I weighted all of the beliefs for each particle and multiplied out a variable using the emission model similar to what I did in number 3. I then covered the second edge case where all of the particles could be weighted as 0. If this was the case then I reinitialized the gameState using initializeUniformly and if it wasn't I looped through all of the particles and set their value to be the a sample of the beliefs vector. For initializeUniformly I looped through all of the particles and added positions based on the state of the particle list. More detail is available on how this was done is available in the comments of my code. For getBeliefDistribution I just incremented probability distribution value for each particle and then normalized and returned it.

5)
For number 5 our only task was to implement the elapseTime function for the ParticleFilter class. This method was pretty straightforward, I just looped through all of the particles and called the function that the comments in the code told me to call and appended the return value of the function to a new distribution list. The old particles value was then overwritten with the new distribution

6)
This was definitely the most difficult question on the assignment. The first function that I wrote was the initializeParticles function. The first thing that I did was use the tools mentioned in the comments to get a shuffled list of the possible positions for each ghost. From here I looped through all of the particles and the positions and appended positions to the particles list. The getBeliefDistribution code turned out to be the same as for the regular particle filter question. The observeState function was the final thing that I implemented for question six. This was the one that I had the most trouble with for this assignment. I started by dealing with the edge cases mentioned in the comments. The first thing that I did was check if all of the states values were zeros. If they were then I had two for loops for the particles and the number of ghosts

where I would initialize the particles and then change the particle value if the ghost was in jail using one of the functions that we were given already. If not the I would loop through all of the particles and and obtain build a new set of values for the particles which I would use to overwrite the initial self.particles. I then backtracked and handled the general case where I would loop through all of the particles and all of the ghosts and use the emission model to generate a new value for the particle if the ghost was not in jail.

7)
For here what we are doing is kind of similar to what we were doing with the elapseTime for the particle filter class. Since we are dealing with multiple ghosts now we now must loop over every ghost as well as every particle in out distribution. We then will use the line of code given in the comments to get a new distribution. Once we have the new distribution all we have to do is sample it and the skeleton that we were given pretty much does the rest.