CS440 PART 2

1. In WorldWithoutThief

(a) We set the discount factor to .9, the learning rate to .1, and the epsilon to 0.0 and what we saw was that the Agent after the first episode (which is almost always high negative reward) decides that getting rewards of zero is better than getting negative rewards. So we see that after the first we have the rest of the episodes incurring rewards of zero. The agent stays in its place or loops because it does not want to have any negative cost! This is because since the Agent will not explore at all, it finds that going to zero is better than negative rewards. Essentially by keeping the epsilon at 0 the Agent stays in its comfort zone (or purely greedy/no exploration).

We also see this in the best policy simulation. The best policy is usually to walk into a wall or stay in a loop giving it a reward of 0 forever. It has not explored its world and therefore only knows that getting 0 is better than slipping.

(b)
After setting the epsilon as .1 we see that the Agent has enough room to explore and finds a way to get very high rewards in the 200s. This is because the epsilon allows the Agent to explore a little bit. So it is greedy but from time to time we make sure that the agent is looking at other options other than the ones it already knows. So the agent eventually will go to every square possible while at the same time figuring out which ones are best unlike in part a where it only knew a piece of the board.

We also see this in the best policy simulation. The agent knows which policies are the best and successfully is able to go back and forth from the company to the customers. When it drops the package it goes back. This ties directly to the fact that the Agent was allowed to explore and find all the best Qvalues possible.

(c)
We chose to set the epsilon value to 0.5 and we see that the values of the rewards are consistently in the negatives. This is because the exploration is too high. The amount of times we tell the Agent to forget the path it knows and go a direction at random is too many! The agent is essentially going random direction way too much and it incurs heavy negative rewards as a result. We also checked the RandomAgent under these same conditions and it produces negative rewards for each episode of around -40. Although the QLearningAgent at .5 epsilon is sometimes above -40 and sometimes below -40 it shows that we are going random too much.

2. In WorldWithThief

(a)
Even with the relatively low epsilon we see that since the Agent incurs heavy negative rewards about -12 per episode! This is because the Agent does not know where the thief is. It doesn't find the best path because it doesn't consider where the thief is and it gets 'surprised' because previous strong moves become absolute failures. Essentially the Agent is told that the board is static but it is not, the thief is moving! Since the board is dynamic we confuse the Agent by making past good steps have a thief on it. We dont tell the Agent where the thief is in its q value table so we can choose to avoid the thief wherever it is (after the agent gets robbed once)!

What it needs to do is if the best move contains the thief we need to take the second best move. This is why the Agent fails drastically when it doesn't know the thief!

(b)
When we know where the thief is we do extremely well. We have rewards near 300 per episode. We are correctly telling the Agent that there is a thief and that we should avoid it. The Agent acts accordingly and finds the best paths by walking around the thief or waiting for it until it moves. It does the opposite of what was explained above and therefore does very very well! It looks at second best choices and avoids the thief at all costs. It's best poliscy as a result does very well when it knows as well. If the amount of steps and episodes are sufficient we get a robot that never touches the thief.

(c)

Let us explore to find the best values for the epsilon and the learning rate. We tried our .05 epsilon and .1 learning rate and it did very well so lets try doing better than that. The average the settings in (a) which did fairly well was approximately 273.0.

We found the estimated max values for the learning rate and the epsilon informally by playing around initially then finding a pattern to increase our average rewards. We did this until no matter what value we chose (and we tried a lot) the average was not greater than the values we found. Below shows how we got to our values.(Note: we found the average reward rate for each execution by adding all the rewards for each episode then dividing them by the number of episodes so of course the average is still an estimate because each run may differ slightly. Therefore whenever I present an average in the data here I took the average of the average of five runs so we can get as close as possible to the real rewards). Below is a small set of data points that showed how I progressed as I changed the value of alpha and epsilon

Average	295.9	301.01	304	308.2
Epsilon	.03	.025	.025	.02
Learning Rate	.2	.2	.15	.11

Average	311	313.0	319.51
Epsilon	.015	.011	.008
Learning Rate	.11	.11	.18

So as we can see after progressing we could not find a higher reward by changing alpha and epsilon via trial and error. So we will choose epsilon = .008 and alpha = .18 as our values for the next problem.

3.

We can see that the learning rate with our first trial is pretty strong and that it plateaus after a certain point early on in the execution. The plot is noisy but we can still see the pattern we expected (first attachment to see the plot or the csv file to understand plot points).

Now let us plot the average of ten runs(see second attachment)! We see here that because we are plotting the averages there is much less noise than the earlier plots. There is almost a clear line of what is going on although the general shape is the same.