1.a

Running simulator with epsilon = 0 results in all episodes getting total rewards of 0, except for the first one. Since epsilon is 0,there is no randomness in choosing the steps. It always choose the next step with highest utility. However in the world setting, both customers are surrounded by slipper blocks, Q learning algorithm won't try to get into any of these slipper blocks with negative rewards since it always perform greedily.

b.

With epsilon = 0.1, the agent experienced some negative total reward in the first few episodes. After a few episodes, it begins to get positive total rewards, ranging from one digit number to a few hundred. With randomness, agent is able to explore through some slipper block and find the positive-rewarded customer block. During the first few episodes, the agent's Qvalue table is quite empty, so it runs into many slipper block and hence get negative total rewards. As the agent runs a few episode, it learned to avoid some slipper blocks as it filled up the Qvalue table. So it can now deliver the packages with the least penalty.

c.

With epsilon = 0.5, most of the episodes results in negative total rewards. This is because of the high randomness in choosing next step to go. The agent did learn through the process and improve by filling the Qvalue table. But the randomness is so

high that it has only 0.5 chance to perform the "correct step" according to the Qvalue table. So it ends up stepping into so many slipper block by randomness.

2.a

All the episodes get negative total rewards. This can be explained by the existence of thief and the fact that agent don't know the position of thief. So it is quite possible that the package will be stolen and result in a negative value in the Qvalue table. Since the thief can move along that column, all the blocks in that column give negative reward. Therefore, in the PolicySimulator, the thief would rather not to deliver the packages than being stole by the thief.

b.

Now with the knowledge of position of the thief, the agent can easily avoid the thief by crossing the column in right timing, and deliver both packages. So the resulting total reward for most of the episodes are positive.

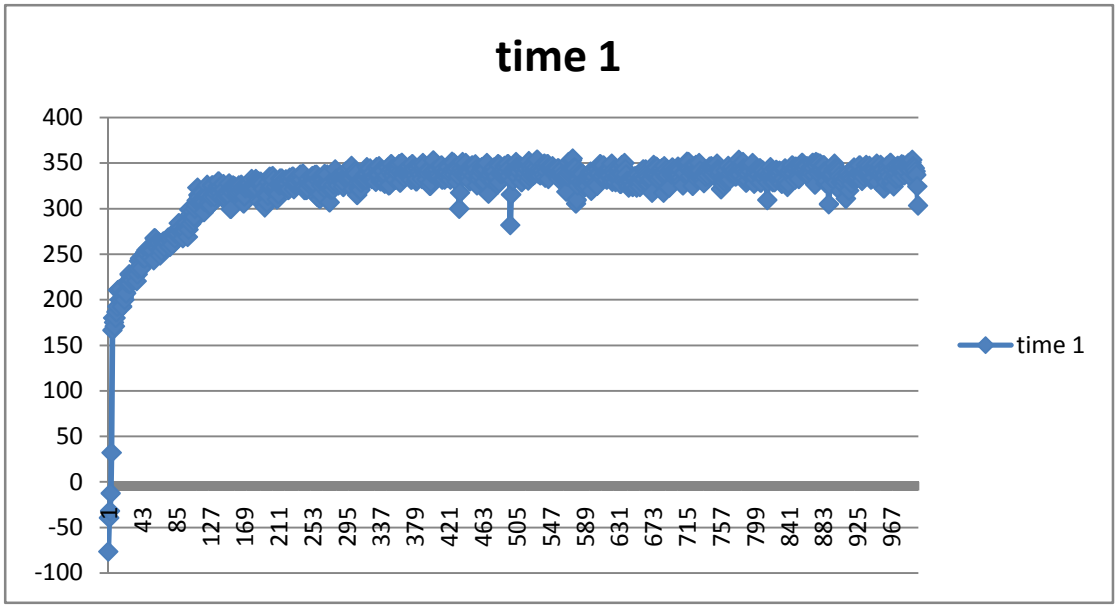
c. The best variables I have found so far are rate = 0.2, epsilon = 0.005. I began comparing the result of fixed rate of 0.1 and different epsilon, and found out that for larger than 0.005, the bigger epsilon the lower the highest total reward. This is quite obvious since the higher you set the randomness the more you go against the best strategy. But somehow for epsilon lower than 0.005, for example, epsilon = 0.0005, the results is quite low and nearly half of the episodes behave like epsilon = 0. This I

think is because the randomness is so low, that the agent don't the chance quite often to explore blocks that "seems to go against" the best strategy so far, and thus learn much slower than the case with higher epsilon.

As for learning rate, I fixed epsilon to 0.005, and tried different rate for 0.1 to 1. It doesn't seemed much different for 0.2< rate <0.5, with a very small tendency of declining in maximum total reward. But for rate >0.5, the declining pattern becomes more clearer. For rate > 0.2, it also appears to be declining. So I guess the peak is at rate = 0.2.

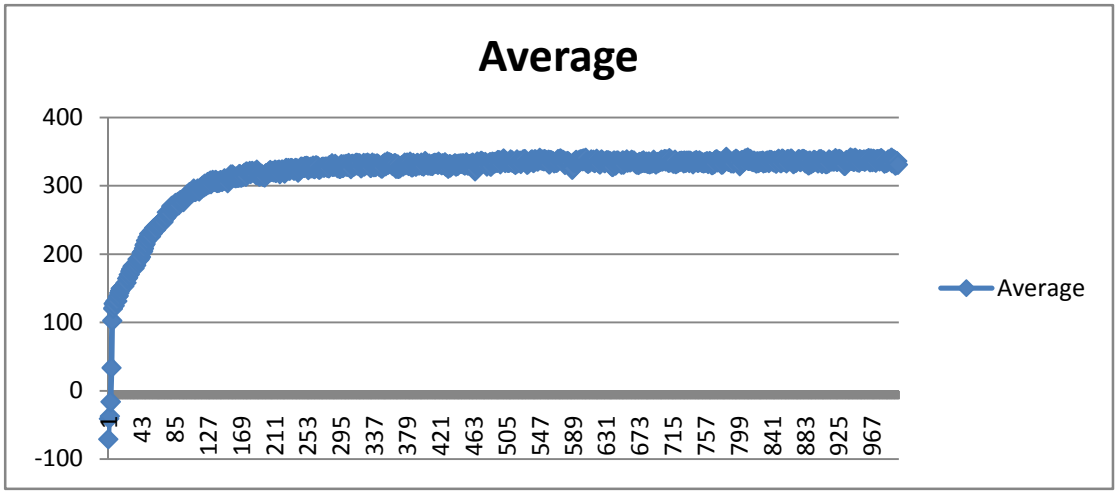
3.

Here is the plot of the first time:



This plot suggests a rapid increase in total reward in the early episodes, and decreasing rate of increase over all. We also can see a pattern fluctuation in the whole graph. The overall shape of this plot quite assemble logarithmic increase.

Here is the plot of an average of 10 times:



The curve is more smooth and now the logarithmic pattern is more clear. This may suggest the learning rate of Q-learning algorithm is logarithmic