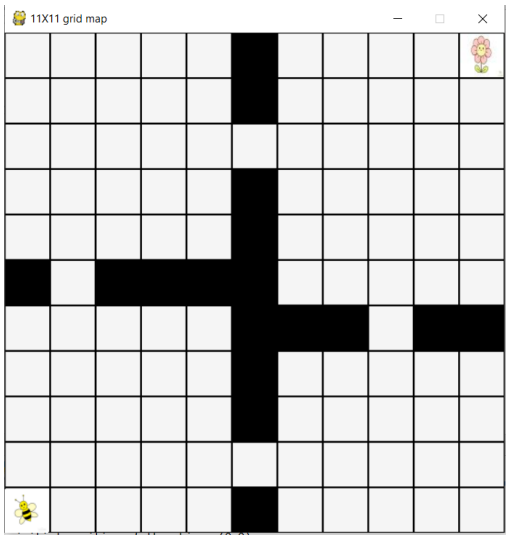
Please refer to the readme_before_use.pdf for running the code.

Answer for question1 and question2:

Run: python pygame_grid.py, it will show you the interface as follows:



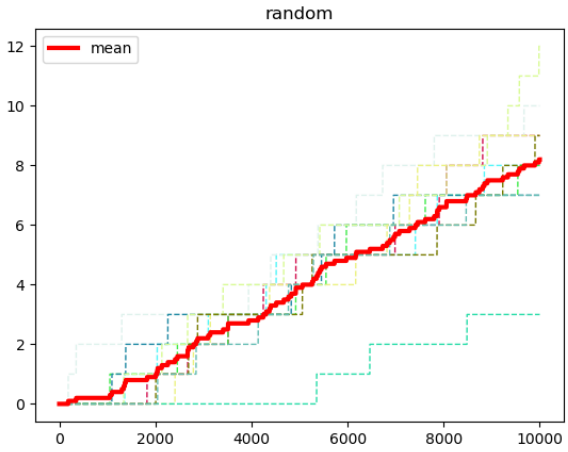
The bee in the figure represents the agent and the flower in the top-right corner is the target. You could use the 'arrows' in the keyboard to navigate the agent. It also prints the action (right, left, up, down) it took actually and the coordinate position.

Answer for q3:

Run: python simulator.py --planer='random'

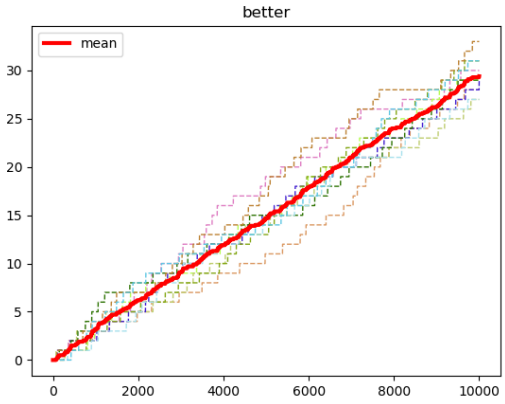
It would play the game automatically for 1e4 steps.

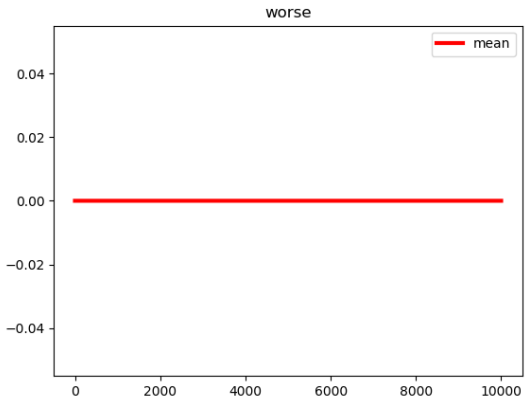
The figure below shows the return it got for 10 runs: the average return is in the range of (8,9)



Answer for q4:

- For better policy: python simulator.py --planer='better'
- For worse policy: python simulator.py --planer='worse'





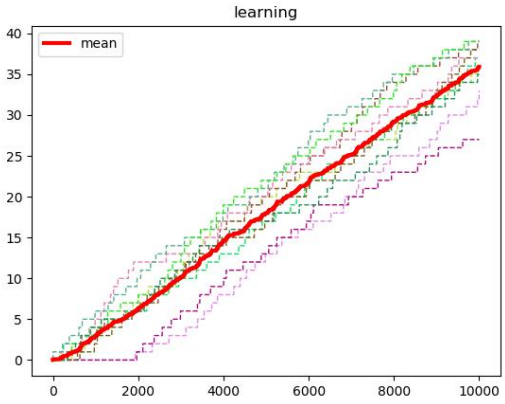
Details about the better and worse policy:

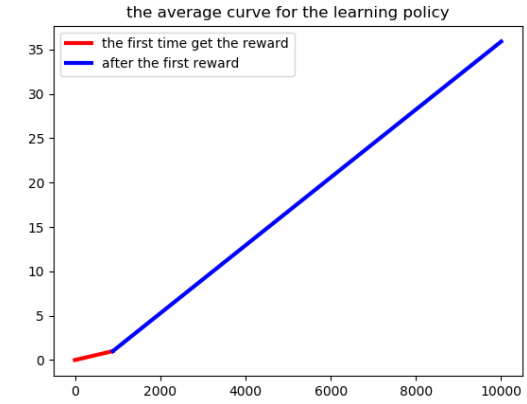
For the target is located at (10,10), increasing the probability of selecting the right action and up action would improve the result. In the better policy, the probability of selecting the right and up action is 0.3 for each, and the probability of selecting the down and left action is 0.2 for each. In the worse policy, it always selected the up action.

Answers for q5:

The goal position was randomly sampled at (7,8) in my test.

- For learning policy: python simulator.py --planer='learning'





It shows in the figure that after the first time it got the target, the agent would find the target more rapidly.

Details about the learning agent:

Inspired by the better policy, the correct direction would lead to high return. With the prior knowledge that the agent is located at (0,0), the target must be located in the top-right direction with respect to the initial agent position. How to distribute the probability to the right action and up action? With the reward signal, the agent would know the goal's position the first time it find the goal. With the goal's position, we could assign the probability to the right action and up action according the horizonal distance and the vertical distance between the goal and the initial agent position. To be specific, I use the pseudo code to describe the process.

1. Ran random policy to find the goal
2. The first time the agent got a reward, it knew the goal's position e.g. (7,8)
3. With the knowledge of the goal's position e.g. (7,8), it would update the policy: selecting the right action with probability of $0.7 * \frac{7}{7+8}$, selecting the up action with probability of $0.7 * \frac{8}{7+8}$, and selecting the left and down action with probability of 0.15, respectively.

It would improve the policy after getting the first reward signal. However, it's a naïve learning policy since it just learned one time. The figure above shows the different slope for the first time it got the reward and after the first reward.