Jeffrey Alberts	3083179
Jur Kersten	4168763
Steven Fleuren	5506425

Project C - Lab-report

Exercise 1

a. The difference between the value iteration agent and the Q-learning agent lies in the information they have access to. The former has access to the transition function and the reward function, and can use them to calculate the optimal policy without taking actions in the gridworld. The latter does not have access to those two functions, but it does know its current state and the actions it can use. After acting in the gridworld it gets to know which state it reached and what reward belongs to transitioning from the last state to the current one by acting like it did. Note that taking the same action does not always lead to the same state because of noise. In short, value iteration is a method for offline learning and Q-learning is a method for online learning without knowledge of the model.
b. The methods that can not be used in reinforcement learning are `getTransitionStatesAndProbs` and `getReward.` The former can be used to construct the transition function, the latter is the reward function. Both need information about the model to return the correct values, and the reinforcement agent does not have access to that information.
c. In the MDP implementation in `gridworld.py` the rewards belonging to the current state are given when the agent leaves this state. The exit states are not terminal states in this implementation. Once the agent reaches an exit state it can only use one action, which takes the agent to the true terminal state. Since the agent leaves the exit state with this action it collects the reward belonging to the exit state. The course-slides use a similar implementation: see slide 28 of the course notes on MDPs.
d. The following definition is given in *Artificial Intelligence: A Modern Approach* by Stuart Russell and Peter Norvig on page 162: "States where the game has ended are called **terminal states**". It is clear that the game has ended once the agent reaches the true terminal state. There is still an action to be taken in the exit state, namely the action that leads to the true terminal state. So the game is not truly over yet and the exit states are no terminal states.

Exercise 2

An offline planner uses information about the model to find a good policy, without actually acting in the world. An online planner has less or no information about the model and has to try out actions in the world to find out which actions lead to high rewards.

Exercise 3

a. The type of task which includes the use of one or more terminal states is called an episodic task. Here the interaction is broken into episodes which terminate in the end state. The MDP which implements this is called an episodic MDP.
b. The need of a discounting factor in continuing tasks is so the infinite horizon expected cumulative reward converges for the algorithm. The inclusion of terminal states makes this convergence happen naturally. Taking a discount closer to 1 means the algorithm will be more farsighted and focus on the end goal. Having a discount very close to 1 means the difference between a discounted and not discounted state is very slim. Since the needed factor in an episodic task is closer to 1 compared to a continuing task we can state that the inclusion of terminal states have the effect that the chosen discount factor will be closer to 1.

Exercise 4

a. The default discount of 0.9 shows that the agent doesn't attempt to cross the bridge from a state near the low reward terminal state.
   We first take the extremes of the discount.
   Using a discount of 1.0 shows the same policy as 0.9. The agent would only attempt to cross the bridge once it starts at the halfway mark between the two terminal states.
   A discount of 0 however makes the agent revert to the low reward state even if it would start right next to the high reward state.
   A discount of 0.1 gives the same policy that 1.0 and 0.9 give us. The agent does not attempt to cross the bridge unless it starts at the halfway mark.
   Discount factor of 5.0 , 3.0 and 7.0 give us differing values but the exact same policy every time. The Agent never attempts to cross the bridge unless starting from the middle state halfway the bridge.
   Reverting back to the default discount we set the noise at extremes. A noise of 1 makes the agent suicidal. It will always choose the exit state north with a high penalty.
   A noise of 0 however make the agent cross the bridge naturally since there is no chance of it diverting from its path.
   Setting the noise at 0.5 shows the same policy we see with 0.2 if different values.
   Choosing a noise of 0.3 or 0.7 gives us no difference in policy. We now know the noise needs to be close to 0 so a noise of 0.1 is tested. It still will not attempt a crossing with those chances, so we change the noise a decimal more.
   0.01 gives a crossing to the higher reward state to the other side of the bridge
b. We chose to change the noise to a value of 0.01. In the previous question we saw that a change in discount did not affect the policy to the desired result. Naturally with a noise of 0.2 the chance at a high penalty state is too high for the algorithm involved. We also saw that we needed to change the noise to get the desired result. A noise of 0 (so no random movement) gives the optimal policy but no noise at all is not a desired factor of this

algorithm. Using trial and error we came upon the highest noise factor that can be used while still facilitating a crossing of the bridge. This only happens if the noise is 0.01 or lower. Since some randomness is better than none we elected to use 0.01 as our value.

Exercise 5

a. For each of the 5 policy types, the tuples <γ (discount), n (noise), r (living reward)> is presented below.
   3.a: < 0.3, 0.2, -6>
   3.b: < 0.3, 0.2, 0>
   3.c: < 0.9, 0.2, -1>
   3.d: < 0.9, 0.2, 0>
   3.e: < 0.9, 0.2, 1.6>
b. To explain the policies and how discount, noise and living reward play a role in this the effects of these three factors first needs to be explained. Discount represents the difference in importance between future rewards and present rewards where γ (discount) has a range between 0 and 1. The higher the discount (so the closer the γ is to 0), the calculation of which action to take will be based more on the present rewards and less on the future rewards. So a policy with a high discount will have a higher focus on fast/close rewards than on slow/long term rewards.
   As for noise, this factor is the change that an action that is taken goes wrong. So, for example the chance that, when the action is to go north the outcome is going to the east or to the west. Like discount, noise also has a range between 0 and 1. In the policies that were chosen, the noise has been kept at a constant 0.2, meaning that the chance that the action does not lead to the desired outcome is 20%.
   Living reward is basically an extra reward system that is given to specific states and actions. It is not always necessary to implement a living reward, and the range of a living reward can be a lot bigger than the ranges of discount and noise and can also be negative. The difference between a living reward and the reward that is received when reaching a goal state is that living reward is given just for performing an action. So, it is not goal-state dependent like the end reward is. When a living reward is sufficiently positive, it is profitable to keep on going from state to state, instead of going to the goal state immediately. When a policy is negative, this works the other way around. There is a punishment for each time an action is taken to another state, therefore the path that leads to a goal state will be kept short and the closest goal state will most likely be chosen. If the negative living reward is too high, it might actually be the case that the policy is to go off the cliff since that can mean a less negative reward than the one being implemented by living reward.
   So, the <u>first policy</u> prefers taking the close exit (+1), but risks the cliff (-10) in getting there. For this, the policy used a high discount (0.3) and a high negative living reward (-6). The high discount causes the policy to focus on the short term rewards more than long term rewards, which causes the choice for the close exit since this is a faster reward than the distant exit even though the reward for the close exit is significantly smaller. This also has to do with the high negative living reward. A policy always strives to get the maximum sum of rewards possible, and when, every time a

step is taken, a negative reward is received the choice for a short path to a quick termination is the best. These factors cause the policy to prefer taking the close exit (quick termination) and risks the cliff (short path), since avoiding the cliff will, thanks to the high negative living reward, cost more than actually falling off the cliff.

The <u>second policy</u> prefers taking the close exit, but avoids taking the route that leads past the cliff. So, it takes a relatively long path to a quick termination. With what has already been said about the involved factors, a logical solution was to keep the same high discount (0.3) since this will make the policy focus on short term rewards over long term rewards. However, in this case the policy had to avoid taking the risky route past the cliff. This could simply be handled by setting the living reward to 0, giving less incentive to take a short route since that route could lead to a very high negative reward when falling off the cliff. With no living reward, it is simply not worth it risking that anymore.

For the <u>third policy</u>, the discount was heightened to the 'standard' discount value of 0.9, and the living reward was set to -1. This policy prefered to take the distant exit (+10) over taking the close exit and risked the cliff in getting there. The fact that the discount value is 0.9 makes the policy prefer long term rewards a lot more than the high discount that was used in the previous policies. This causes the preference for the distant exit. The negative living reward of -1 is pretty low compared to the -6 that was used in the first policy, but considering the discount it is not illogical that the risky route is taken. This since the distant exit is preferred, making the path longer and the amount of steps that have to be taken higher. So the total sum of negative living rewards would still be pretty high taking the safe route. Therefore, it is more profitable for this policy, when trying to maximize the sum of rewards, to take the short but unsafe route.

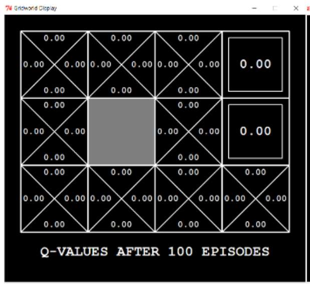
As for the <u>fourth policy</u>, it prefers taking the distant exit (slow termination) and avoids taking the cliff (long/safe route). A similar relation can be found between itself and the third policy as there is between the first and second policy. Where the choice that is made for the distant exit is based on the same discount value as for the third policy, focussing on long term reward over short term reward. The difference between this policy and the third one can be found in the living reward. As in policy 2 it is set to 0, making it less costly to take the long and safer route over passing the cliff since a longer path now costs less because no negative living rewards are added. So, this policy takes the highest reward that takes longer to reach, and takes a safe path since the chance of it going wrong (noise 20%) is not worth the little extra reward that could be gotten.

The <u>fifth policy</u> is quite special, since this policy should lead the episode to never terminate since all goal states (both exits as well as the cliff) are avoided. This is counter-intuitive since a policy always strives to get the maximum sum of rewards, so to reach this the only logical solution is to make the living reward high enough for this to exceed the reward it gets for terminating (either +1 or +10 depending on the exit, in this case +10 since the discount value stays the same at 0.9). It appears that this can be achieved with a living reward of 1.6 (or higher). The policy then becomes to keep on going instead of terminating, since the maximum sum of rewards keeps increasing and exceeds +10 it gains for terminating. Also, it avoids the cliff since it is in the policies best interest to make as many steps as it can for it is rewarded for every step that is taken.

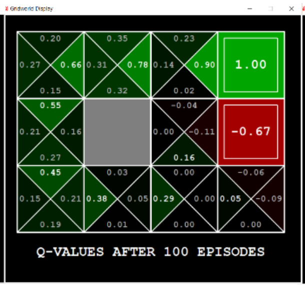
Exercise 6

A. Learning rate (α) can also be named step size, a term that might more simply explain the definition of this term. This since learning rate determines to what extent the information that has been calculated will override the old information. Learning rate has a range from 0 to 1, where a value of 1 means that all the old information will be overridden by the newly calculated values. This means that the agent would be fully deterministic. When the learning rate is 0, the hypothesis is that the agent would not learn anything, since no values would be overridden by the new scores, so all values (which start at 0 in the crawler) would then stay at 0. Various learning rates have been put to the test, using 100 iterations, a standard discount and noise rate (0.9 and 0.2 respectively), no living reward and an exploration rate of 0.1 (randomness in step-taking). The different outcomes are presented below.

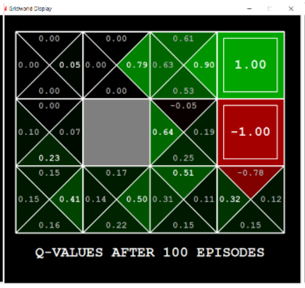
I: Learning rate: 0.0 Average return from start state: -0.0679072155961
II: Learning rate: 0.2 Average return from start state: 0.386381650213
III: Learning rate: 0.5 Average return from start state: 0.159623234135
IV: Learning rate: 0.8 Average return from start state: 0.27660729346
V: Learning rate: 1.0 Average return from start state: 0.30019740831



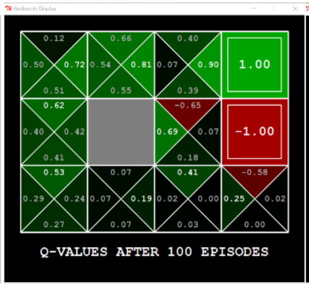
I



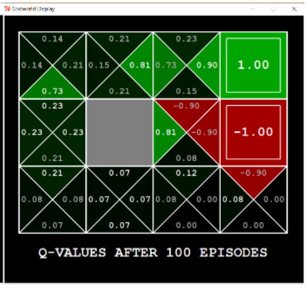
II



III



IV

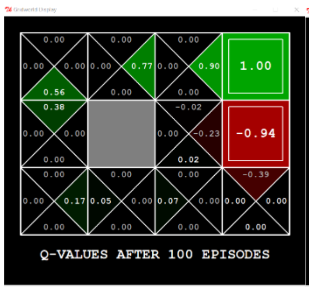


V

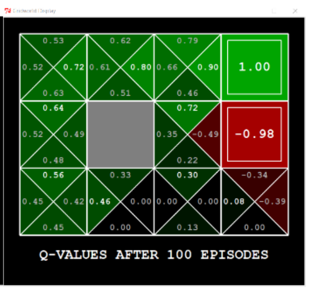
As hypothesized, with a high learning rate, the Q-values are very dependent on the most recent value and can therefore be changed very drastically when a certain action from one state to another is taken. This high learning rate does find the safest path to the goal rather quickly, since the rewards (also negative rewards) for actions are high and almost immediate. This does however mean that, when a wrong path is chosen (thanks to noise or randomness for instance) the agent tries to go back to the start state rather than press on. And the scores can very easily be somewhat 'reset' when the agent does not find the goal state quickly. As has been hypothesized about a low learning rate, at a rate of 0 nothing will be learned, so learning is impossible. At a learning rate of 0.2, learning is possible but it is a lot slower than at a high learning rate. So to conclude, setting a high learning rate means that a path will be learned rather quickly. But, this has it's consequences since it gives no guarantees that the path that is found at first (which in most cases is the path that will get the highest Q-values) is actually the best path to take. So quick learning comes at the cost of some possible inaccuracy. A low learning rate will take longer to optimize/converge but is more accurate.

B. Exploration rate is the amount of randomness that is added to the agent in its decision to choose a particular path. It must be distinguished from the noise, since this can also be interpreted as a certain degree of randomness but these are different factors. Where noise is the chance that, when an action is chosen it doesn't succeed and another action is taken the exploration rate has effect on which action is chosen in the first place. With a range between 0 and 1, where 0 meaning that no randomness is added and 1 meaning every action that is taken is chosen arbitrarily (or at random). Exploration rate is added so that, from time to time (depending on the exploration value of course), an action is chosen that does not necessarily have to be the best (considering the Q-values) in order to not overlook any paths to the goal that would otherwise be overlooked because their values are 0. This because other actions could have a higher value simply because those have been randomly taken first. In this experiment all factors remained the same as in the experiment above with the exception of learning rate, which has been set at 0.5. So the only varying factor is the exploration rate.

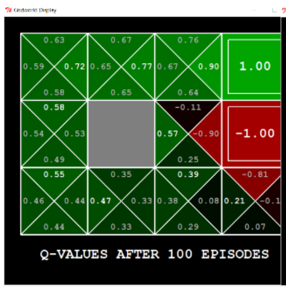
I: Exploration rate: 0.0 Average return from start state: 0.0588948411587
II: Exploration rate: 0.2 Average return from start state: 0.361931766613
III: Exploration rate: 0.5 Average return from start state: 0.174897949351
IV: Exploration rate: 0.8 Average return from start state: -0.0111392334692
V: Exploration rate: 1.0 Average return from start state: -0.0713633391857



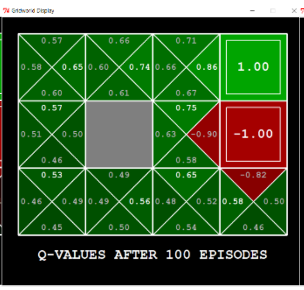
I



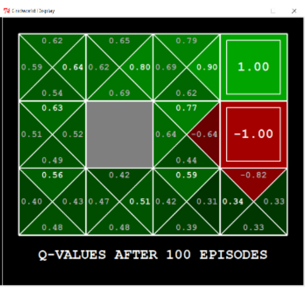
II



III



IV

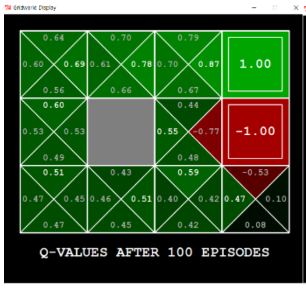


V

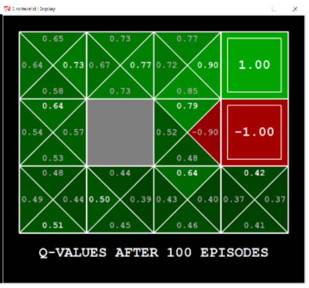
At a low exploration rate, it looks like the agent is prone to getting 'stuck' between two states, when the actions that lead from the one state to the other and back have high Q-values respective to the other actions in that state. Also, a lot of actions don't yet have calculated Q-values (Q-value is equal to 0). With a higher exploration rate (even at 0.2), a lot more calculations have been made (a lot less black in picture II compared to I). The randomness makes seems to make sure that every possible action is calculated, which also seems to result in a good conversion. A setback of a higher exploration rate is that, as with all randomness, if it gets too high it will not necessarily find the best path since it is sort of all over the place. It just takes all paths across all possible states and the results will not be very high (as can be seen in the average return for IV and V. To conclude, a certain amount of exploration is needed in order to find the best path, but a high exploration rate comes at the cost of directness and speed.

C. When combining both learning and exploration rate, one can hypothesize about the effect that these factors have on each other. It is likely that, when adopting a low learning rate and a high exploration rate, this will make the agents actions and policy very random and it will most likely take a long time for the agent to converge and find the best path. This is because the agent will inherit a lot of randomness from the exploration rate, whilst not learning very fast (getting a relatively small reward) for reaching the goal state. This results in Q-values that are relatively close together within states. When the Learning rate is heightened, together with a high exploration rate, this effect is heightened as well. The high learning rate encourages the agent to find a quick path, and with the high exploration rate this path is often changed. In the end, this can result in quick and accurate pathfinding very quickly but it can also change scores pretty significantly pretty fast which can result in some weird scores (see the Q-values of the start state in figure II for instance). Low rates for both learning and exploration, in comparison to the high rates, seems to be more goal directed, it has lower values in general (which is to be expected) and seems to favour one path specifically (see figure III). This leads to an agent that seems pretty effective in finding the goal state. It does however have very low values for the negative rewards gotten at the negative goal state and it's surrounding q-values. This leads to believe that a combination of low learning rate and low exploration rate makes an agent prone to risk taking behaviour, since when it has found its positive goal state a couple of times (because of the low learning rate), it will not quickly vary from the route it takes to get there because of the limited exploration done. This does also lead to a rather high average return (compared to I and II). A high learning rate and a lower exploration rate seems to lead to an agent (agent IV) that is more risk-avert than agent III. Where it can reach the goal state, by taking an action to the east, it tries to go to the north instead. This seems to be because the agent is frightened to (because of the noise) make the action south and thereby getting closer to the negative goal state. It is also, by far, not converged yet. Q-values close to the goal states are pretty low. When both learning rate and exploration rate have a value of 0.5, Q-values seem to normalise across the states, seemingly making the policy pretty extensive and not necessarily direct. This seems that way because actions that don't necessarily seem to contribute to reaching the goal state still have a pretty high Q-value (like the Q-value of going west in the upper left corner in figure V). It is still pretty safe, considering the best possible action when the wrong action is taken in the start state the agent will try to return to this start state. But, when proceeding too far into that path, the agent calculates that the best course of action is to continue on that path, even though it takes the risk of ending up in the negative goal state.

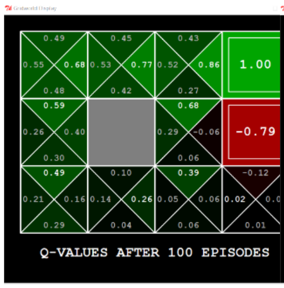
I: Learning rate: 0.2 Exploration rate: 0.8 Average return: 0.0701148685471
II: Learning rate: 0.8 Exploration rate: 0.8 Average return: 0.00592642174484
III: Learning rate: 0.2 Exploration rate: 0.2 Average return: 0.357218845745
IV: Learning rate: 0.8 Exploration rate: 0.2 Average return: 0.193763920488
V: Learning rate: 0.5 Exploration rate: 0.5 Average return: 0.191508624736



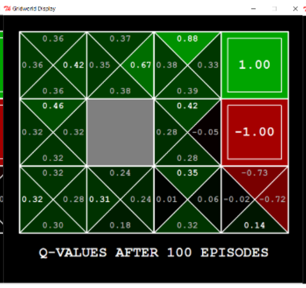
I



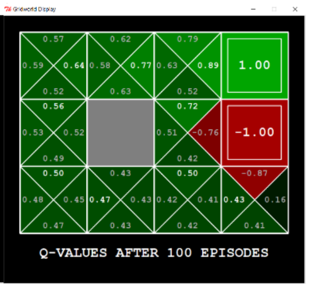
II



III



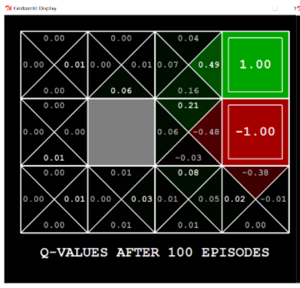
IV

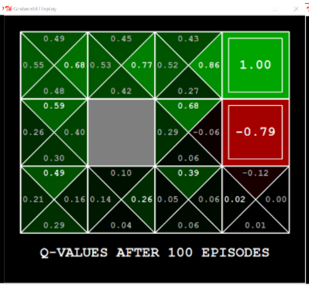


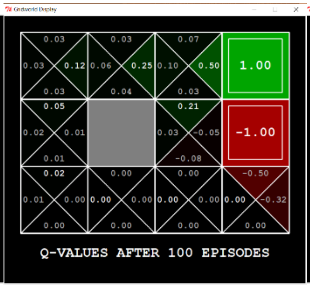
V

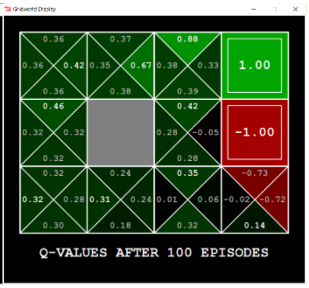
D. Yes, the effect of changing the learning rate does depend on the value of the discount rate. Even though the immediate value that is given to reaching the goal state is not affected by the discount and thus is only dependent on the learning rate it does follow through in the Q-values for reaching these states. These are lowered exponentially when the discount rate is high (γ = 0.5). This is because the Q-value is the product of discount and the reward in the next state (in the state next to the goal state). So when the value of the goal state is 0.2 (in this case after reaching the goal state once), the Q value for going west from the state next to the goal state will only be 0.1. And after each in the state that is next to that it will only be 0.025. So, even though the value that is given to the goal state (which depends on the learning rate) does not directly depend on the discount, the calculated Q-values are affected by the discount and will therefore be smaller when the discount value is high (closer to 0). This will cause the agent to take a very long time before it converges (difference in effect can clearly be seen in figures I and II). For figure I, a lot more episodes (or iterations) need to be done before the values are calculated enough to make a clear path from start state to goal state. But then, the question is does the changing of the learning rate and its effect depend on the value of the discount rate. It looks like this is the case because of the exponential element that is in the calculation of the Q-values. Where for each iteration, when the Q-value is calculated, the discount rate is exponentiated by itself. So the Q-value gets exponentially smaller every time because of the discount rate. So the lower the initial discount rate is, the lower the discount value of each state and thus the (exponentially) lower the Q-value will become.

I: Learning rate: 0.2 Discount rate: 0.5 Average return: 0.0016908957541
II: Learning rate: 0.2 Discount rate: 0.9 Average return: 0.357218845745
III: Learning rate: 0.8 Discount rate 0.5 Average return: 0.00741733771487
IV: Learning rate: 0.8 Discount rate 0.9 Average return: 0.193763920488



I


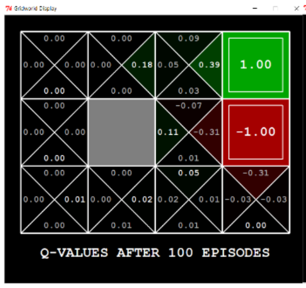
II


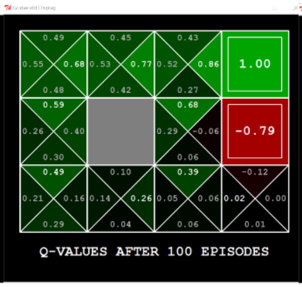
III


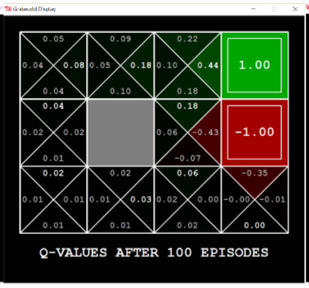
IV

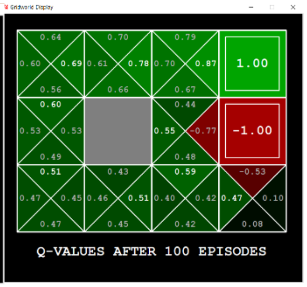
E. Unlike learning rate and its direct effect on Q-values, exploration rate does not have such a direct effect on the Q-values. Its effect is more indirect in the fact that it influence path that is chosen and randomise the chosen path, ignoring the Q-values in that specific instance. This does however have an effect on the newly calculated Q-values, when the Q-value is received after this action was taken. The effect that exploration rate has is that the higher the exploration rate, the more the Q-values are normalised (close together per state). This effect is not changed by the discount rates. The values will still be normalised, especially close to the start state (this can be seen in the figures below). The fact that the, because of the discount, the Q-values are a lot lower has effect on the states and the final path (depending on how many iterations are done) and the exploration rate has an effect on how the path will turn out and how the agent will act. These two factors however don't influence each other, or have an effect that strengthens or diminishes each other's effects. That would also be strange, seeing as mathematically these two factors are not intertwined, where learning rate and discount are.

I: Exploration rate: 0.2 Discount rate: 0.5 Average return: 0.000724274665262
II: Exploration rate: 0.2 Discount rate: 0.9 Average return: 0.357218845745
III: Exploration rate: 0.8 Discount rate 0.5 Average return: -0.000414440254815
IV: Exploration rate: 0.8 Discount rate 0.9 Average return: 0.0701148685471



I


II


III


IV

Exercise 7

a. In the given model there are no living rewards and there is no noise, so for each state the best action will be to either move west or east. The reward in the western exit state is 1, and the reward in the eastern exit state is 10. Suppose the agent is one tile to the east of the western exit state. Moving one tile west gives a reward of 1 * discount = 0.9. Moving one tile east could only be the best action if moving east is the best action in the other non-exit states as well, otherwise the agent would not reach an exit state when acting under the optimal policy and not collect a reward. The reward the agent would get when starting at the west side of the bridge and crossing the bridge is 10 * discount^5 = 5.9049 > 0.9. So the optimal policy is to always move east. Applying Q-learning with the given input values does not return this policy. The crawler never reaches the eastern exit state, so the reward of that exit state does not affect the Q-values of the explored states. This results in the Q-values of moving east being lower than the Q-values of moving west in each explored state.

b. We have experimented with the values 0.1, 0.3, 0.5, 0.7, and 0.9. The higher values were more likely to find the correct Q-values for the exit states on the left side of the bridge. This is because with a high learning rate new experiences are weighted more heavily, and in the exit states the rewards will always be the same. So the agent has to visit those states less often for them to converge to the correct value. However, the agent is unlikely to reach the right side of the board, so the positive exit state at the east end of the bridge will not affect any of the Q-values, regardless of what learning rate was chosen.

c. We have experimented with the epsilon values 0.2, 0.4, 0.6, 0.8 and 1, and kept the learning rate fixed at 0.5. For low epsilon values, the agent would often move to the western exit state. This is explainable: Once the agent discovers that moving to the west from the starting position gives a positive reward, it will move west with odds 1 - epsilon + epsilon / 4 from the start state in every next episode. This means that for lower epsilon, the agent is less likely to reach the right side of the bridge before the iteration is complete. Even for epsilon = 1 the odds of reaching the right side of the bridge are minimal.

d. With only 50 iterations, the odds of the agent reaching the other side of the bridge are nihil, regardless of which values for alpha and epsilon are chosen. If more iterations are allowed, the odds of reaching the eastern reward state are largest if epsilon = 1. Just reaching this state once is not enough, the agent will have to reach it multiple times for all the values to converge. This process will go more quickly with a high learning rate, since a larger portion of the reward will be added to the Q-value. However, the learning rate should be lowered later in the process to avoid noise having a big impact on the results.

Exercise 8

a. For our experiment, we will observe what happens if we alter one of the constants and keep the others at their default value. We will let the agent play 2000 training games and 100 test games afterwards. The average score of those 100 games will be used as a benchmark for the performance of the agent. The average score of the agent when using the default parameters is 499.79
    - Alternating alpha: We have tested the values 0.1, 0.4, 0.8 and 1. The resulting score for the first value is 490.19 and the agent won 99 of the 100 test games. The difference with the score from the default parameters is not significant, however the lost game indicates that the Q-values have not yet converged to the correct values, which might be because the learning rate is to low. For the value 0.4 the score was 499.39, which is a bit lower than the default score but not significantly so. For 0.8 the score was 500.24, so the agent can still learn a good policy even with a high learning rate. This can be explained by the fact that bad states (Pacman encounters a ghost) are heavily punished. So even if such state is encountered early in the training period the information won't be dominated by new information easily, and the low exploration rate reduces the odds of encountering the situation again. For alpha = 1 we still get a good score of 500.64, but had the agent made a risky move that went well near the end of the training period this could have been much lower.
    - Alternating epsilon: We have tested the values 0 and 0.2. For value 0, the average score was 499.84. It makes sense that the agent still learns to avoid the ghost: If taking an action in a state leads to a collision with a ghost, the agent will never use that action in that state again. However, Pacman might learn to take routes that are longer than necessary because once it finds a good route it won't look for a better one. For the value 0.2 the average score is 399.42, which means that Pacman often runs into a ghost during the training period and never reaches some of the states.
    - Alternating gamma: We have tested the values 0.5 and 0.95. For the first value, the average score is 499.56. A lower discount should lead to the agent focussing on shorter routes, but that does not become clear from this result. This could be caused by the low epsilon value, or maybe the score of the default values is already close to optimal. The average score for the value 0.95 is 499.98. Overall, the effect of changing the discount factor seems to be minimal.

b. There are five things that can be different in each state: The position of Pacman, the position of the ghost, whether or not each food dot has been eaten, and the direction the ghost is facing. The last part is relevant because the ghost will never turn around unless if he reaches a dead end. After each action, the position of Pacman changes one tile (or he walks into a wall or stands still) depending on which action was taken. If the new tile contains a food dot, it will be eaten. The movement of the ghost depends on its current orientation. If it has a choice between moving straight ahead and making a turn it chooses randomly. If Pacman moves into the ghost's current position, the ghost does not move. This means that each action can lead to 1 or 2 (depending on the original position of the ghost) possible new states. The game reaches an exit states if both food dots have been eaten, or if Pacman's position is the same as the position of the ghost. Each action (including standing still) has a reward of -1, with the exception of the action that leads from the exit state to the terminal state. That action rewards either -500 or +500 depending on which exit state was reached. In addition to those rewards, each time Pacman moves into a position with a food dot the agent gets a reward of +10.

c. We trained the PacmanQAgent for 5000 games with the parameters epsilon = 0.05, gamma = 1 and alpha = 0.3. The average reward over all training games is -437.54. We used 100 test games, with a 15/100 win rate and an average score of -355.35. The number of possible states is much larger in this grid compared to the small grid. There is also more open space, which makes the ghost less predictable. Even with the increased number of iterations, many states will not be thoroughly explored, which leads to Pacman walking into ghosts regularly.

Exercise 9

Autograder scores:

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Total
6/6	1/1	5/5	5/5	3/3	1/1	1/1	0[1]	22/22