Q1)

EX 3

1) a) for stochastic case: $v^*_{\pi^*}(s) = \sum_a \pi^*(a|s)\, q^*_{\pi^*}(a,s)$

for deterministic case: $v^*_{\pi}(s) = \max_a q^*_{\pi}(a,s)$

b) $q^*(s,a) = \sum_{s',r} p(s',r|a,s)\,[r + \gamma v^*(s')]$

c) $\pi^*(a|s)$: $\arg\max_a q^*_{\pi}(s,a)$

or $a^* = \arg \pi^*(a^*|s) = \arg\max_a (s,a)$

d) $\pi^*(a|s)$: $\arg\max_a \sum_{s',r} p(s',r|s,a)[r+\gamma v^*(s')]$

or $a^* = \arg \pi^*(a^*|s) = \arg\max_a \sum_{s',r} p(s',r|s,a)[r+\gamma v^*(s')]$

e) $v_{\pi}(s) = \sum_a \pi(s,a)\left[r(s,a) + \gamma \sum_{s'} p(s'|s,a)\, v_{\pi}(s')\right]$

$v^*_{\pi^*}(s) = \sum_a \pi^*(s,a)\left[r(s,a) + \gamma \sum_{s'} p(s'|s,a)\, v^*_{\pi^*}(s')\right]$

$q_{\pi}(s,a) = r(s,a) + \gamma \sum_{s'} p(s'|s,a) \sum_{a'} q_{\pi}(a',s')\, \pi(a'|s')$

$q^*_{\pi^*}(s,a) = r(s,a) + \gamma \sum_{s'} p(s'|s,a) \sum_{a'} \pi^*(a'|s')\, q^*_{\pi^*}(s',a')$

$p(s'|s,a) = \sum_r p(s',r|s,a)$ $\quad r(s,a) = \sum_{r,s'} r\, p(s',r|s,a)$

Q2)

2)a)In policy Improvment part, Instead of comparig new policy and old policy for each state, we should compary value function for both new and old policies like below

⋮

3. Policy Improvement

policy-stable $\leftarrow$ true

For each $s \in S$

$\pi_{old}(s) \leftarrow \pi_{new}(s)$

$\pi_{new}(s) \leftarrow \arg\max_a \sum_{s',r} p(s',r|s,a)[r+\gamma \bar{V}(s')]$

If $\sum p(s',r|s,\pi_{new}(s))[r+\gamma \bar{V}_{\pi_{new}}(s')] > \sum p(s',r|s,\pi_{old}(s))[r+\gamma \bar{V}_{\pi_{old}}(s')]$,

then policy-stable $\leftarrow$ false

If policy_stable, then stop and return V^* and π^*;

else go to polig evaluation part

In this way we never get stuck in an inifint loop

2) b) Yes It is possible to occure also for Value Iteration.
when in episodic tasks the rewards all are positive so $\bar{V}(s)$ will always Increase and agent never goes to terminal state so we need to consider negative reward in this case or using discount factor.
It is also more likley in non episodic task when there is no discount factor (exactly like Q4-Part C) so we need discount factor or generally we could use limit number of iteration.

Q3)

3.3) a)

1. Initialization
   $Q(s,a) \leftarrow R \times R$, $\forall s \in \mathcal{S}$, $\forall a \in A(s)$ arbitrarily
   except $Q(s_{terminal}, a) = 0$

2. Policy Evaluation
   Loop:
      $\Delta \leftarrow 0$
      Loop for each $s \in \mathcal{S}$
      Loop for each $a \in A(s)$
         $q \leftarrow Q(s,a)$
         $Q(s,a) \leftarrow \sum_{s',r} p(s',r|s,a)[r + \gamma \max_{a'} Q(s',a')]$
         $\Delta \leftarrow \max(\Delta, |q - Q(s,a)|)$
   until $\Delta < \theta$

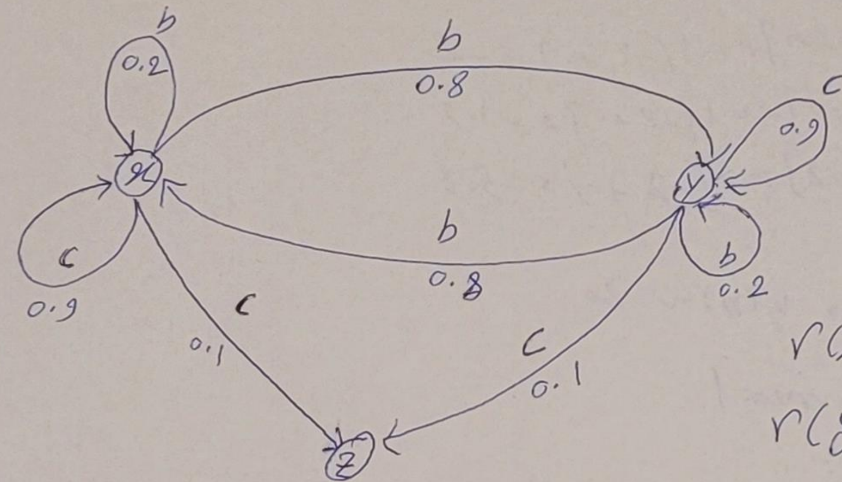
3. Policy Improvment
   policy_stable $\leftarrow$ True
   For each $s \in \mathcal{S}$
      old_action $\leftarrow \pi(s)$
      $\pi(s) \leftarrow \arg\max_a Q(s,a)$
      If old_action $\neq \pi(s)$, then policy_stable $\leftarrow$ False

If policy stable, then stop and return V^*, π^*; else go to 2

Q 3-b and 4)

3) b) $Q_{k+1}(s,a) = \sum_{s',r} p(s',r|s,a)\left[r+\gamma \max_{a'} Q_k(s',a')\right]$

4) a)



$r(x,a) = -1$

$r(y,a) = -2$

In $S=x$, we never want to go to $S=b$ in which for each action we will have -2 reward. in adition, we want to reach $S=z$ which is terminal state as soon as posible. Therefor, in $S=x$ the best action is 'c'.

In $S=Y$, although the Probability of reaching $S=z$ for each action is the same as $S=x$, the reward for each action is -2, double comparing with reward in $S=x$. So, we prefer to go to $S=x$ and try to go to $S=z$ in $S=x$ instead of $S=y$ So the best action in $S=Y$ is 'b'

4) b) Initialization:

$\bar{V}(x)=0$, $\bar{V}(Y)=0$, $\bar{V}(Z)=0$ (always), $\gamma=1$

value evaluation:

$V_1(x)=0.9[-1+0]+0.1[-1+0]=-1$

$V_1(Y)=0.9[-2+0]+0.1[-2+0]=-2$

$V_2(x)=0.9[-1-1]+0.1[-2+0]=-1.9$

$V_2(Y)=0.9[-2-2]+0.1[-2+0]=-3.8$

⋮

$V_i(x)\sim-10$, $V_i(y)\sim-20$

Policy Improvment:

in $s=x$:

$\pi(x): c \Rightarrow \bar{V}(x)=0.9[-1-10]+0.1[-1+0]=-10$

$\pi(x): b \Rightarrow \bar{V}(x)=0.8[-1-20]+0.2[-1+10]=-19$

in $s=Y$:

$\pi(y): c \Rightarrow \bar{V}(Y)=0.9[-2-20]+0.1[-2+0]=-20$

$\pi(Y): b \Rightarrow \bar{V}(Y)=0.8[-2-10]+0.2[-2+20]=-14$

$\pi(x) \leftarrow \arg\max_a \bar{V}(x) = c$

$\pi(y) \leftarrow \arg\max_a \bar{V}(y) = b$

do again value evalution till

$V(x)\sim-10$, $V(Y)\sim-12.75$

Policy Improvment

$s=x$

$\pi(x): c \Rightarrow \bar{V}(x)=-10$

$\pi(x): b \Rightarrow \bar{V}(x)=-13.2$

$S=Y$:

$\pi(y): c \Rightarrow V(Y) = -13.475$

$\pi(y): b \Rightarrow V(Y) = -12.75$

$\pi(x) \leftarrow \arg\max_a V(x) = c$

$\pi(y) \leftarrow \arg\max_a V(y) = b$

$\pi_{old}(x) = \pi_{new}(x)$, $\pi_{old}(Y) = \pi_{new}(Y)$

Stop and return $\pi^*(x): c$, $\pi^*(y): b$

$V^*(x) \approx -10$, $V^*(y) \approx -12.75$

4)

c) If the initial policy for both x, y states is b so we will have

$V(x) = -1 + 0.2V(x) + 0.8V(Y)$

$V(Y) = -2 + 0.8V(x) + 0.2V(Y)$

which are inconsistent so there will be no solution for these equations.

with having discount these two equations will not be inconsistent so we will find a unique answer.

However, very small γ cause we have not optimal policy. For example in this question small γ will merge to 0 in α so cause we select action 'b' even in $S=Y$, which is not optimal policy

For Questions 5 and 6:

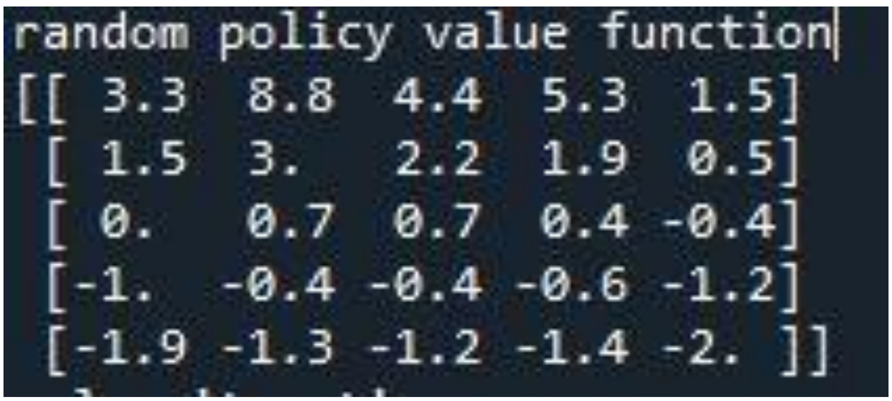
- Both the gridworld and car rental environments are defined in `env.py`.

- The algorithms related to Q5 and Q6 are saved as `Q5_Gridworld.py` and `Q6_JacksCarRental.py`, you just need run these two py files

Q5)

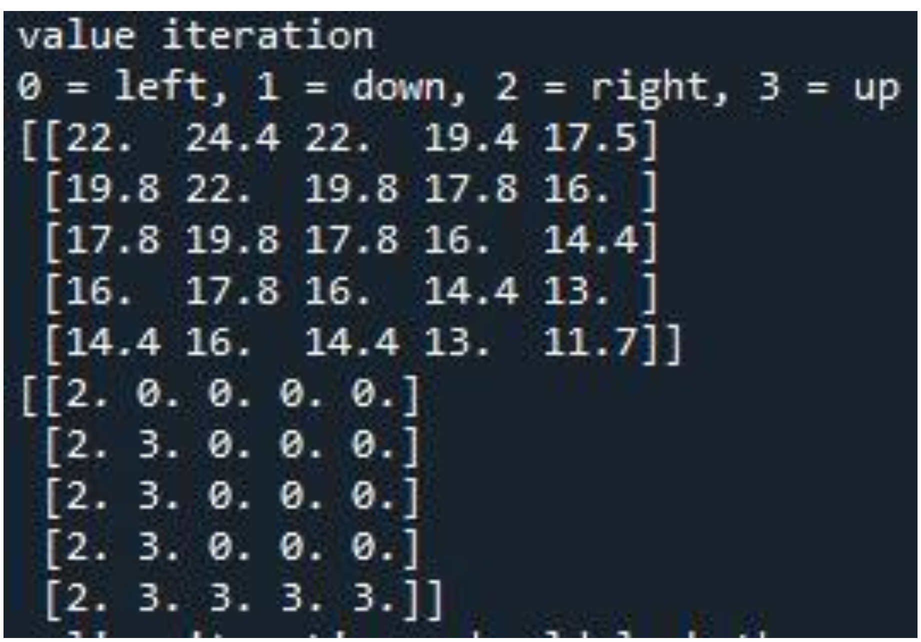
A) ***Iterative policy evaluation***

The value function associated with the random policy. As you can see it matches figure 3.2



```
random policy value function
[[ 3.3  8.8  4.4  5.3  1.5]
 [ 1.5  3.   2.2  1.9  0.5]
 [ 0.   0.7  0.7  0.4 -0.4]
 [-1.  -0.4 -0.4 -0.6 -1.2]
 [-1.9 -1.3 -1.2 -1.4 -2. ]]
```

B) ***Value iteration***



```
value iteration
0 = left, 1 = down, 2 = right, 3 = up
[[22.  24.4 22.  19.4 17.5]
 [19.8 22.  19.8 17.8 16. ]
 [17.8 19.8 17.8 16.  14.4]
 [16.  17.8 16.  14.4 13. ]
 [14.4 16.  14.4 13.  11.7]]
[[2. 0. 0. 0. 0.]
 [2. 3. 0. 0. 0.]
 [2. 3. 0. 0. 0.]
 [2. 3. 0. 0. 0.]
 [2. 3. 3. 3. 3.]]
```

C) ***Policy iteration***

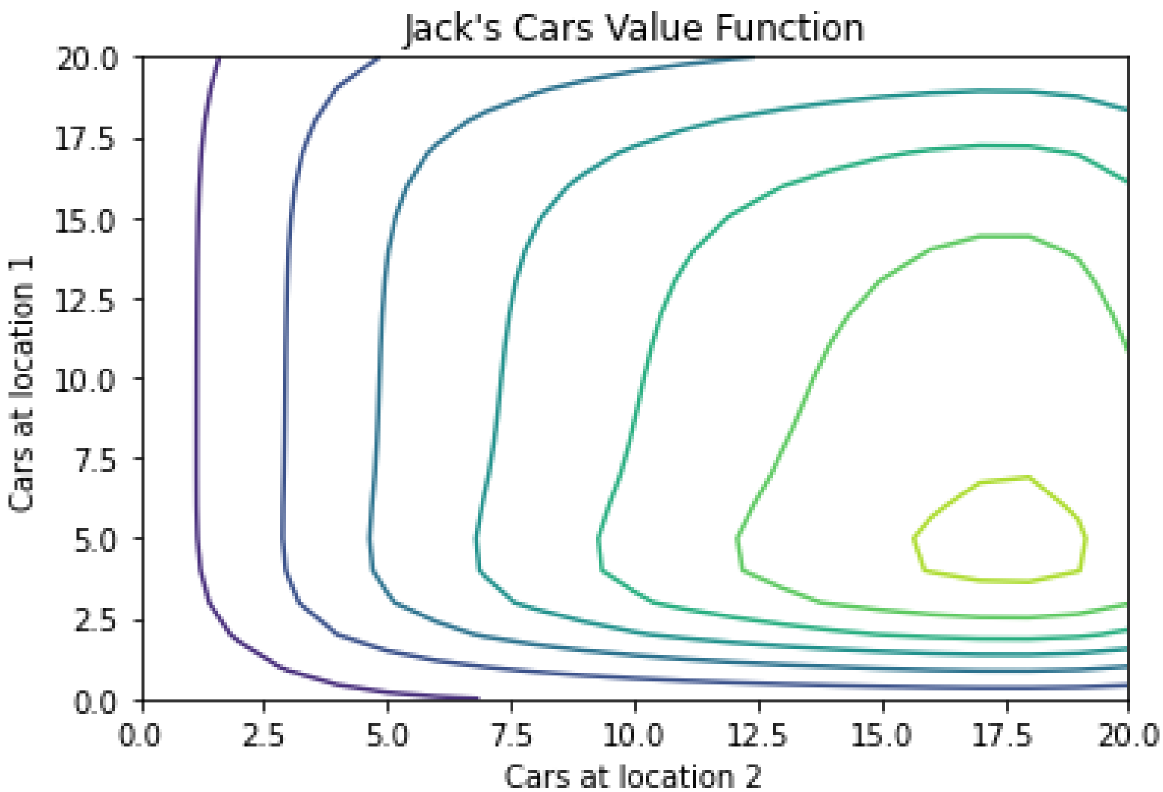
```
policy iteration, should look the same as value iteration
0 = left, 1 = down, 2 = right, 3 = up
[[22.  24.4 22.  19.4 17.5]
 [19.8 22.  19.8 17.8 16. ]
 [17.8 19.8 17.8 16.  14.4]
 [16.  17.8 16.  14.4 13. ]
 [14.4 16.  14.4 13.  11.7]]
[[2. 0. 0. 0. 0.]
 [2. 3. 0. 0. 0.]
 [2. 3. 0. 0. 0.]
 [2. 3. 0. 0. 0.]
 [2. 3. 3. 3. 3.]]
```

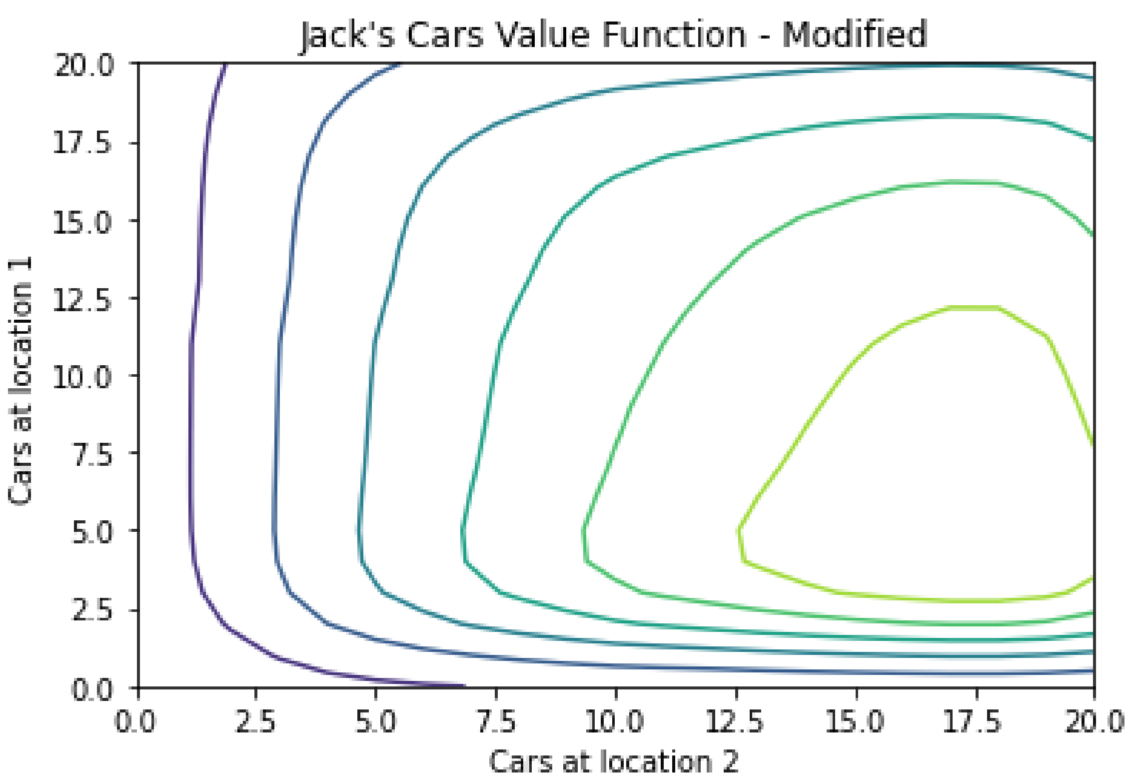
The optimal policy and the associated value function, which match figure 3.5

Q 6)

6. A. Unfortunately, because of an issue in my code, my policy converges to “0” for every state (no movement). So, there is no point at seeing a policy contour map. I provide the final value maps instead, which must also be incorrect.

B. Although I could not produce the correct policy, I could guess that because one piece of movement from lot 1 to 2 is free, the policy will be more willing to move when lot 2 is low. Additionally, we will see more movement in the 6-14 car midrange as we attempt to avoid the 4 dollar parking fee.


Jack's Cars Value Function
Cars at location 1
Cars at location 2
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0


Jack's Cars Value Function - Modified
Cars at location 1
Cars at location 2
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0

Q7)

7) a) $|\max_a f(a) - \max_a g(a)| \leq \max_a |f(a) - g(a)|$

case 1: $\max f(a) \geq \max g(a)$

$a_f = \arg\max_a f(a)$ $\quad a_g = \arg\max g(a)$

$g(a_f) \leq g(a_g)$

$|\max f(a) - \max g(a)| = f(a_f) - g(a_g) \leq f(a_f) - g(a_f)$

$|\max f(a) - \max g(a)| \leq f(a_f) - g(a_f)$

definition of max

$f \geq g$ $\quad f(a_f) - g(a_f) = |f(a_f) - g(a_f)| \leq \max_a |f(a) - g(a)|$

$\Rightarrow |\max_a f(a) - \max_a g(a)| \leq \max_a |f(a) - g(a)|$

for case 2: $\max g(a) \geq \max f(a)$

It would be exactly similar to case 1 with this difference that we need to change the role $f(a)$ and $g(a)$ and we will find

$|\max_a g(a) - \max_a f(a)| \leq \max_a |g(a) - f(a)|$

7) b) $\|BV_i - BV_i'\|_\infty \leq \gamma \|V_i - V_i'\|_\infty$

$\|BV_i - BV_i'\|_\infty = \|\max_a \sum P(s', r \mid s, a)[r + \gamma V_i(s')] -$

$\max_a \sum P(s', r \mid s, a)[r + \gamma V_i'(s')]\|_\infty$

$= \|\max \gamma V_i(s') - \max \gamma V_i'(s')\|_\infty \leq$ based on part a

$\gamma \max \|V_i(s') - V_i'(s')\|_\infty$

$\gamma \max \|V_i(s') - V_i'(s')\|_\infty \leq \gamma \|V_i - V_i'\|_\infty$

$\Rightarrow \|BV_i - BV_i'\|_\infty \leq \gamma \|V_i - V_i'\|_\infty$