Question 1

	s	a	s'	r	p(s', r | s, a)
1	high	search	high	r_{search}	α
2	high	search	low	r_{search}	$1-\alpha$
3	low	search	high	-3	$1-\beta$
4	low	search	low	r_{search}	β
5	high	wait	high	r_{wait}	1
6	low	wait	low	r_{wait}	1
7	low	recharg	high	0	1

$$p(s', r \mid s, a) = Pr\{S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a\}$$

The function indicates that there is a probability of the values $s' \in S$, $r \in R$ occurring at a particular time given the previous state & reward.

From the table we can see that for each combination of (s, a) there is a unique (s', r) given s' i.e. for each (s, a) there is only one possible successor state with only one possible reward.

We can see from the table and the problem description that given s, a, s', there is only one possible r, or, equivalently

$$p(s' \mid s, a) = p(s', r \mid s, a)$$

$$p(s, r \mid s, a) = p(s' \mid s, a) \cdot p(r \mid s', s, a)$$

where $p(r \mid s', s, a) = 1 \quad \forall (s', s, a)$

Question 2

~~3.15~~ 3.15

The signs of the rewards are not important, the relative difference between the rewards are important as ~~the signs~~ are just a cons.

(3.8) → $G_t = R_{t+1} + \gamma R_{t+2} \cdots \sum_{k=0}^{\infty} \gamma^k R_{t+1+k}$

adding a constant 'c' to all the rewards affects G_t in the following manner

$$G'_t = (R_{t+1} + c) + \gamma(R_{t+2} + c) + \gamma^2(R_{t+3} + c) \cdots$$

$$= \sum_{k=0}^{\infty} \gamma^k (R_{t+1+k} + c)$$

$$= \sum_{k=0}^{\infty} \gamma^k R_{t+1+k} + c\sum_{k=0}^{\infty} \gamma^k$$

$$= G_t + \frac{c}{1-\gamma}$$

it affects $v(s)$ as follows

$$V'_s(s) = E[G'_t \mid S_t = s]$$

$$= E\left[G_t + \frac{c}{1-\gamma} \mid S_t = s\right]$$

$$= E[G_t \mid S_t = s] + E\left[c/1-\gamma \mid S_t = s\right]$$

$$= E[G_t \mid S_t = s] + \frac{c}{1-\gamma}$$

$$V'_t(s) = V_t(s) + \frac{c}{1-\gamma}$$

∴ It does not affect the relative value of any state under any policy

$$Vc = \frac{c}{1-\gamma}$$

3.16

For an episodic task,

$$G_t = \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$$

adding a constant c to each R

$$G'_t = \sum_{k=t+1}^{T} \gamma^{k-t-1} (R_k + c)$$

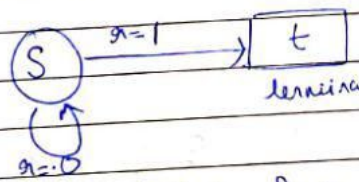
$$= \sum_{k=t+1}^{T} \gamma^{k-t-1} (R_k) + c \sum_{k=t+1}^{T} \gamma^{k-t-1}$$

$$G'_t = G_t + c\left(\frac{1-\gamma^T}{1-\gamma}\right)$$

∴ it adds a ~~constant~~ term, as above, however the term depends on T

It does impact the task. as the value added is not constant

For example,



goal is to reach terminal state as fast as possible

If there is no addition of c to each reward, the agent can terminate at t=1 to maximize reward.

If we add c to each reward, the longer the agent stays at S, the more reward it reaps, the total reward converging to $\frac{c}{1-\gamma}$ for larger T ($\to \infty$)

and this would affect the task as the agent would not reach the terminal state as fast.

Question 5

We know that $V_*(s)$ is the maximum value that can be achieved from an MDP starting at state s. →

$$V_*(s) = \max_{a \in A(s)} q_*(s, a)$$

In order to have the optimal value at s, $V_*(s)$, we must take the action which gives us the best value from the MDP, $q_*(s, a^*)$. a^* is the action that corresponds to the best action which gives $V^*(s)$. (found by maximum over all possible actions from s).

Question 6:

A few iterations of PI:
0.00 -7.83 -11.12 -12.23
-7.83 -10.42 -11.77 -11.86
-11.12 -11.77 -11.05 -8.81
-12.23 -11.86 -8.81 0.00
evalutation count 10
0.00 -11.43 -16.30 -17.93
-11.43 -14.84 -16.57 -16.61
-16.30 -16.57 -15.11 -11.84
-17.93 -16.61 -11.84 0.00
evalutation count 20
0.00 -12.93 -18.46 -20.30
-12.93 -16.68 -18.57 -18.59
-18.46 -18.57 -16.79 -13.10
-20.30 -18.59 -13.10 0.00
evalutation count 30
0.00 -13.55 -19.36 -21.29
-13.55 -17.45 -19.40 -19.41
-19.36 -19.40 -17.50 -13.62
-21.29 -19.41 -13.62 0.00
evalutation count 40
0.00 -13.81 -19.73 -21.71
-13.81 -17.77 -19.75 -19.75
-19.73 -19.75 -17.79 -13.84
-21.71 -19.75 -13.84 0.00
evalutation count 50
0.00 -13.92 -19.89 -21.88
-13.92 -17.90 -19.90 -19.90
-19.89 -19.90 -17.91 -13.93
-21.88 -19.90 -13.93 0.00

A few iterations of VI:
[[0. 1. 1. 1.]
[1. 1. 1. 1.]
[1. 1. 1. 1.]
[1. 1. 1. 0.]]

[[0. 0. 0. 0.]
[0. 0. 0. 0.]
[0. 0. 0. 0.]
[0. 0. 0. 0.]]

```
[[ 0. -1. -1. -1.]
 [-1. -1. -1. -1.]
 [-1. -1. -1. -1.]
 [-1. -1. -1.  0.]]

[[ 0. -1. -2. -2.]
 [-1. -2. -2. -2.]
 [-2. -2. -2. -1.]
 [-2. -2. -1.  0.]]

[[ 0. -1. -2. -3.]
 [-1. -2. -3. -2.]
 [-2. -3. -2. -1.]
 [-3. -2. -1.  0.]]
```

Solved the bug by taking a deterministic argmax or by comparing sets.

Question 4:

In this case, we need to solve a system of non-linear equations. We can do this either through policy iteration, value iteration, or by treating it as an optimization problem.
In order to treat it as an optimization problem, we model each of 4 possible equations we have to maximize over in the form of AX>=B where A becomes (100,25),B becomes (100,1) in comparison to the linear case where we solve AX=B with A (25,25) and B is (25,25).