Question 1

Part (A)

The state space is

$$\mathcal{S} = \{0, 1, 2, \dots, 100\}.$$

State 0 is the starting state and state 100 is the terminal state.

Part (B)

For any state $s < 100$ and die roll $k \in \{1, 2, 3, 4, 5, 6\}$,

$$P(s' \mid s) = \frac{1}{6}.$$

If $s + k$ corresponds to a snake or ladder, the transition is deterministic to its destination. If $s + k > 100$, the transition moves to state 100. The terminal state satisfies $P(100 \mid 100) = 1$.

Part (C)

```
from rl.markov_process import FiniteMarkovProcess

snakes = {16: 6, 48: 26, 49: 11, 56: 53, 62: 19,
          64: 60, 87: 24, 93: 73, 95: 75, 98: 78}
ladders = {1: 38, 4: 14, 9: 31, 21: 42, 28: 84,
           36: 44, 51: 67, 71: 91, 80: 100}

transition_map = {}

for s in range(101):
    if s == 100:
        transition_map[s] = {100: 1.0}
    else:
        transitions = {}
        for die_roll in range(1, 7):
            next_pos = s + die_roll
            if next_pos > 100:
                next_pos = 100
            elif next_pos in snakes:
                next_pos = snakes[next_pos]
            elif next_pos in ladders:
                next_pos = ladders[next_pos]
            transitions[next_pos] = transitions.get(next_pos, 0) + 1/6
        transition_map[s] = transitions

mp = FiniteMarkovProcess(transition_map)
traces_generator = mp.traces(start_state=0)
game_lengths = [len(trace) - 1 for trace in traces_generator(1000)]
```

Question 2
Part (A)
The optimal value function is

$$V^*(s) = \frac{9}{4} \quad \text{for all } s \in \mathcal{S}.$$

The optimal action is

$$a^* = \frac{1}{4}.$$

Part (B)
We know the optimal action is

$$a^* = \frac{1}{4}.$$

This action does not depend on the state s. Therefore, the optimal deterministic policy for all states is

$$\pi^*(s) = \frac{1}{4}.$$

Part (C) Answer

The optimal value function
The optimal value function will rely on the states, and will be strictly decreasing as s increases, due to the decreasing action space present in the game, especially as you near the ending.

The structure of the optimal policy
The optimal policy is deterministic and selects the maximum action (the one that returns the max value)

Question 3
Part (A)
State space:

$$\mathcal{S} = \{0, 1, 2, \ldots, n\},$$

where 0 is the absorbing failure state (snake) and n is the absorbing success state (escape).
Action space:

$$\mathcal{A} = \{A, B\}.$$

Transition function: for $1 \le i \le n-1$,

$$P(i-1 \mid i, A) = \frac{i}{n}, \qquad P(i+1 \mid i, A) = \frac{n-i}{n},$$

and for action B,

$$P(j \mid i, B) = \frac{1}{n} \quad \text{for all } j \in \{0, 1, \ldots, n\} \setminus \{i\}.$$

States 0 and n are absorbing.
Reward function:

$$R(s) = \begin{cases} 1 & \text{if } s = n, \\ 0 & \text{otherwise.} \end{cases}$$

Part (B)

```
MDPRefined = dict
def get_lily_pads_mdp(n: int) -> MDPRefined:
    data = {
        i: {
            'A': {
                i - 1: i/n,
                i + 1: (n-i)/n,
            },
            'B': {
                j: 1/n for j in range(n+1) if j != i
            }
        } for i in range(1, n)
    }
    data[0] = {}
    data[n] = {}

    gamma = 1.0
    return MDPRefined(data, gamma)

Mapping = dict
def direct_bellman(n: int) -> Mapping[int, float]:
    vf = [0.5] * (n + 1)
    vf[0] = 0.
    vf[n] = 1.
    tol = 1e-8
    epsilon = tol * 1e4
    while epsilon >= tol:
        old_vf = [v for v in vf]
        for i in range(1, n):
            va = (i/n) * vf[i-1] + ((n-i)/n) * vf[i+1]
            vb = sum(vf[j] for j in range(n+1) if j != i) / n
            vf[i] = max(va, vb)
        epsilon = max(abs(old_vf[i] - v) for i, v in enumerate(vf))
    return {v: f for v, f in enumerate(vf)}
```

Part (C) Answer

The patterns I observed that the optimal policy does affect how the frog behaves based on pond size. For small ponds like size 3 and 5, basically only the lily pad right next to the snake (lily pad 1) chooses the random jump strategy, while all the other lily pads go with local movement since they're pretty safe. But for larger ponds like 10, 15, 25, there is higher risk because the danger zone expands where there is more lily pads near the snake and the froggies decide random jumping is safer.

When the frog is on lily pad 13 the action it should take is action B (random jump).
This action is the same as what the frog does on lily pad 1 where both spots choose action B.

Question 4 Part (A)

$$v_0(s_1) = 10, \quad v_0(s_2) = 1, \quad v_0(s_3) = 0.$$

For $k = 1$:

$$q_1(s_1, a_1) = 11.15, \quad q_1(s_1, a_2) = 11.40, \quad v_1(s_1) = 11.40, \quad \pi_1(s_1) = a_2,$$
$$q_1(s_2, a_1) = 4.15, \quad q_1(s_2, a_2) = 2.05, \quad v_1(s_2) = 4.15, \quad \pi_1(s_2) = a_1,$$
$$v_1(s_3) = 0.$$

For $k = 2$:

$$q_2(s_1, a_1) = 13.5475, \quad q_2(s_1, a_2) = 12.80, \quad v_2(s_1) = 13.5475, \quad \pi_2(s_1) = a_1,$$
$$q_2(s_2, a_1) = 5.0425, \quad q_2(s_2, a_2) = 4.1325, \quad v_2(s_2) = 5.0425, \quad \pi_2(s_2) = a_1,$$
$$v_2(s_3) = 0.$$

Part (B)
For state s_2, action a_1 is already better than a_2 after the first iteration, and this ordering never changes in later iterations. This is because both actions depend on the same future state values, and those values only increase together as value iteration proceeds.
For state s_1, after the second iteration we have

$$q_2(s_1, a_1) > q_2(s_1, a_2).$$

In later iterations, both action-values increase as the value function updates, but in this policy we have it choose a_1 over a_2. Since the terminal state value is fixed and the updates use the same transition probabilities each time, the action that is better at iteration $k = 2$ remains better for all subsequent iterations.
Therefore, the greedy policy does not change after the second iteration, and

$$\pi_k = \pi_2 \quad \text{for all } k > 2.$$

Part (C)
Using $\pi_2(s_1) = a_1$ and $\pi_2(s_2) = a_1$, we get

$$V^{\pi_2}(s_1) \approx 16.838, \quad V^{\pi_2}(s_2) \approx 7.119, \quad V^{\pi_2}(s_3) = 0.$$

Part (D)
With $R(s_1, a_2) = 11$, $v_0(s_1) = 10$, $v_0(s_2) = 1$, $v_0(s_3) = 0$, one value-iteration update gives

$$q_1(s_1, a_1) = 11.15, \quad q_1(s_1, a_2) = 12.40,$$

so the greedy choice at s_1 is a_2 after one iteration.

Second part of the question:
This change impacts the optimal deterministic policy a_2 and becomes the maximizing action in the end compared to a_1, as value is greater given that action. So in subsequent state, assuming there was one, we'd begin with a_2

Question 5

Question 1: False. Increasing $R(s, a)$ by 2 increases the immediate term by 2, but Q_π can change by more than 2.

Question 2: True. discounted return is

$$G_0 = 5 + 0.9 \cdot 3 + 0.9^2 \cdot 1 = 8.51 > 6.$$

Question 3: True. For all $\gamma \in [0, 1)$, B_π is a contraction under $\| \cdot \|_\infty$, so it has a unique fixed point, per fixed theorem

Question 4: False. Policy improvement has a possibility where next policy is the same as the previous.

Question 5: True. If $Q_\pi(s, a) = 10$ for all a, then

$$V_\pi(s) = \sum_a \pi(a \mid s) \, Q_\pi(s, a) = 10$$

for any policy π.

Question 6: True. If rewards are bounded and $\gamma < 1$, then discounted return converges to a finite value.