(a) \( L = \{ u u^R v \mid u, v \in \langle 0 \rangle^* \} \) is regular, since
\( L \) is defined by a regular expression
\[
\langle 0 \rangle^* 0 u 1 \langle 0 \rangle^* 1 .
\]
To see that \( L \subseteq L(\langle 0 \rangle^* 0 u 1 \langle 0 \rangle^* 1) \):
Let \( x \in L \) be arbitrary, i.e., \( x = u u^R v \) for some \( u, v \in \langle 0 \rangle^* \).
Since \( u \in \langle 0 \rangle^* \), \( u = 0 u' \) or \( u = 1 u' \) for some \( u' \in \langle 0 \rangle^* \).
Assume \( u = 0 u' \) (the case for \( u = 1 u' \) is similar).
Then \( x = u u^R v = 0 u' v (u')^R = 0 u' v u^R 0 \in \langle 0 \rangle^* 0 \).
To see that \( L(\langle 0 \rangle^* 0 u 1 \langle 0 \rangle^* 1) \subseteq L \):
Let \( x \in L(\langle 0 \rangle^* 0 u 1 \langle 0 \rangle^* 1) \) be arbitrary.
Assume \( x \in \langle 0 \rangle^* 0 u \) (the case for \( x \in 1 \langle 0 \rangle^* 1 \) is similar).
Then \( x = u u^R v \) where \( u = 0 \) and \( v \in \langle 0 \rangle^* \),
that is, \( x \in L \).

(b) \( L = \{ u u^R v \mid u, v \in \langle 0 \rangle^+ \} \) is not regular. Suppose that it was.
We can apply the Pumping Lemma directly on \( L \). Here we use closure properties for regularity first to "restrict" \( L \) to \( L' \):
Consider \( L' = L \cap \langle 0 \rangle^* 0 1 \langle 0 \rangle^* 0 1 \).
Certainly \( L' \) would be regular since "\( \cap \)" preserves regularity.
But, what is \( L' \) (or, why do we consider "\( \cap \)\( 1\langle 0 \rangle^* 0 1 \langle 0 \rangle^* 0 1 \)"
odd number of \( 0 \)s)?

For \( x \in L' \), \( x \) is of the form \( u u^R v \):
Hence \( L' = \{ 10^{2n+1} 110^{2n+1} 11 \mid n \geq 0 \} \).

Now, apply the Pumping Lemma on \( L' \) (remember, there are many cases to check).
b. "On input string w:
   1. Scan the tape and mark the first 0 which has not been marked.
      If there is no unmarked 0, go to stage 5.
   2. Continue scanning and mark the next unmarked 0. If there is
      not any on the tape, reject. Otherwise, move the head to the
      front of the tape.
   3. Scan the tape and mark the first 1 which has not been marked.
      If there is no unmarked 1, reject.
   4. Move the head to the front of the tape and repeat stage 1.
   5. Move the head to the front of the tape. Scan the tape for any
      unmarked 1s. If none, accept. Otherwise, reject."

c. "On input string w:
   1. Scan the tape and mark the first 0 which has not been marked.
      If there is no unmarked 0, go to stage 5.
   2. Continue scanning and mark the next unmarked 0. If there is
      not any on the tape, accept. Otherwise, move the head to the
      front of the tape.
   3. Scan the tape and mark the first 1 which has not been marked.
      If there is no unmarked 1, accept.
   4. Move the head to the front of the tape and repeat stage 1.
   5. Move the head to the front of the tape. Scan the tape for any
      unmarked 1s. If none, reject. Otherwise, accept."

As mentioned in homework statement: give its
formal tuple definition (Q, E, S, ...)
and brief interpretations for the states/ transitions.
Problem 6: We show below that there exists a (deterministic) Turing machine that recognizes $A \cup B$, but the proof/construction is "existential" — no effective way to give formal definition of the machine.

For the finite language $B$, we can decompose $B$ into disjoint union of $B \cap A$ and $B - A$.

Notes:
1. We do not claim that there is an effective computation for partitioning $B$ into the two subsets $B \cap A$ and $B - A$ (but these two subsets do exist), and
2. Both parts/subsets $B \cap A$ and $B - A$ are finite since $B$ is.
3. Since $A$ is Turing-recognizable, denote its recognizer as $M_A$.

Consider the following description of a (deterministic) Turing machine $M$:

On input $w \in \Sigma^*$, $M$:
1. If $w$ is in the finite set $B \cap A$, then $M$ rejects (as halts) $w$.
2. If $w$ is in the finite set $B - A$, then $M$ accepts (as halts) $w$.
3. Simulate $M_A$ on input $w$:
   - accept, reject, run forever as $M_A$ does.

We argue that $L(M) = A \cup B$ as follows. Consider the four disjoint possibilities:
For \( w \in \Sigma^* \):

Case when \( w \in B \setminus A \): According to Step 2 of \( M \),
\[ \text{\( M \) rejects (or halts) \( w \) correctly.} \]

Case when \( w \in B - A \): According to Step 2 of \( M \),
\[ \text{\( M \) accepts (or halts) \( w \) correctly.} \]

Case when \( w \in A - B \): Since \( w \notin B(=B\setminus A) \cup (B - A) \), \( M \) enters Step 3
to simulate \( M_A \) on \( w \).
\[ \text{As \( M_A \) accepts \( w \), \( M \) accepts (or halts) \( w \).} \]

Case when \( w \notin \Sigma^* -(A \cup B) \):

Since \( w \notin B \), \( M \) enters Step 3
to simulate \( M_A \) on \( w \).
\[ \text{As \( M_A \) rejects \( w \) or runs forever on \( w \),} \]
\[ \text{\( M \) rejects \( w \) or runs forever on \( w \) accordingly.} \]

Thus, we can see that \[ L(M) = (B - A) \cup (A - B) \].

Elaborating (1):

The decomposition of \( B \) into \( B \setminus A \) and \( B - A \) is existential.
We may construct \( 2^{18} \) machines (automata), each of which corresponds to a possibility of \( B \setminus A \) (as a subset of \( B \)) and its recognition.

One of these \( 2^{18} \) machines can be used to "implement" the computations in Steps 1 and 2 of \( M \) — but there is no effective algorithm to determine which one (out of \( 2^{18} \) candidates).

Also, the finiteness of \( B \) (hence of \( B \setminus A \) and \( B - A \)) is important in the "implementation" of the computations/decisions in Steps 1 and 2 of \( M \).
5. A Turing machine $T$ that decides $\text{DISJ}_{\text{DFA}}$:

On input $w$ to $T$:

1. Check if $w = \langle A, B \rangle$ for DFA's $A$ and $B$:
   - "yes": go to step 2
   - "no": $T$ rejects/halts on $w$.

2. Construct a DFA $C$ from DFA's $A$ and $B$ such that $L(C) = L(A) \cap L(B)$ via Cartesian product of the state-spaces of $A$ and $B$.

3. $T$ simulates the decider for $E_{\text{DFA}}$ on input $\langle C \rangle$.

   - If the decider accepts/halts on $\langle C \rangle$,
     - Then $T$ accepts/halts on $w$.
   - Else $T$ rejects/halts on $w$.

Check:

1. $L(T) = \text{DISJ}_{\text{DFA}}$ or

2. $T$ halts on every input.
Problem 6. Similar to the machine constructed in the proof of textbook Theorem 4.22.

Denote the TMs recognizing A and B by \( T_A \) and \( T_B \), respectively. Construct the following TM \( T \):

On input \( w \):
1. \( \text{Number of steps} := 1 \).

2. \( T \) simulates \( M_A \) on input \( w \) for \( \text{Number of steps} \) steps:
   
   [If \( M_A \) accepts/halts on \( w \) during the simulation, then \( T \) accepts/halts on \( w \)].

3. \( T \) simulates \( M_B \) on input \( w \) for \( \text{Number of steps} \) steps:
   
   [If \( M_B \) accepts/halts on \( w \) during the simulation, then \( T \) rejects/halts on \( w \)].

4. \( \text{Number of steps} := \text{Number of steps} + 1 \).
   go to Step 2.

Claim 1. The language accepted by \( T \), \( L(T) \), satisfies that \( A \cap B \subseteq L(T) \) and \( A \cap \overline{B} \subseteq L(T) \), or
   
   2. \( T \) halts on every input.