1. **True, false, or nonsense.** For each of the subproblem below, read the statement very carefully, and decide if the statement is true, false, or nonsense. A statement is nonsense if the sentence cannot be parsed, either because there is a type mismatch, or the objects are not well-defined. You don’t need to justify your answers.

Each correct answer gets 1 point; each incorrect answer gets —0.5 points; not answering the subproblem and leaving it blank gets 0 points. (Which means, guessing the answers uniformly at random receives 0 points on average. Yes, the total score can be negative.)

Let $L$ and $L'$ be two arbitrary languages.

(1) Every regular language has infinite size.

**Solution:** False. The empty language $\emptyset$ is regular and has size 0.

(2) Every string in a regular language has finite length.

**Solution:** True. By definition, every string in every language has finite length.

(3) If every string in $L$ is accepted by some DFA $M$, then $L$ is regular.

**Solution:** False. Any single string is accepted by some DFA. But the union of such strings might not be accepted by the same DFA.

(4) If every string in $L$ is regular, then some DFA $M$ exists that accepts $L$.

**Solution:** Nonsense. Strings cannot be regular.

(5) If some DFA $M$ exists that rejects all strings in $L$, then $L$ is regular.

**Solution:** False. There is a single-state DFA that rejects all strings, and thus rejecting all strings in any language $L$.

(6) If $L$ is regular, then some DFA $M$ exists that rejects all strings in $L$.

**Solution:** True. Consider the one-state DFA that rejects all strings.

(7) If $L \cup L'$ is regular, then both $L$ and $L'$ are regular.

**Solution:** False. Consider a non-regular language $L$ and its complement $\Sigma^* \setminus L$. Their union $\Sigma^*$ is regular.

(8) If $L^*$ is regular, then $L$ is also regular.

**Solution:** False. Consider any non-regular language over $\{0, 1\}$ together with two extra strings $0$ and $1$. $L^*$ is equal to $\Sigma^*$ thus regular.

(9) If both $L$ and $L'$ are regular, then $L \setminus L'$ is also regular.

**Solution:** True. Regular languages are closed under complement and intersection, and $L \setminus L' = L \cap (\Sigma^* \setminus L')$.

(10) If $L \subseteq L'$ and $L$ is not regular, then $L'$ is also not regular.

**Solution:** False. Take $L' = \Sigma^*$.

(11) Any language accepted by an NFA has a regular expression.

**Solution:** True. First turn the NFA into an DFA using subset construction, then into a regular expression through the state-elimination algorithm.

(12) Regular expressions are regular.
Solution: Nonsense. A regular expression is not a language, thus cannot be regular. It is true that regular expression themselves are strings over alphabet \( \Sigma \cup \{ (, ), +, * \} \); but then it is the set of regular expressions that is a language, which is not regular (because checking if the parentheses are balanced is not).

(13) Any NFA can be turned into an equivalent DFA recognizing \( L \).

Solution: False. Not any NFA, unless its language is \( L \).

(14) Any regular expression of \( L \) can be turned into an equivalent NFA.

Solution: True. Union, concatenation, and Kleene star are all regular-preserving operations and thus can be implemented using NFAs.

(15) Any language has a fooling set.

Solution: True. Not necessarily infinite in size.

(16) If \( L \) has a finite fooling set, then \( L \) is regular.

Solution: False. Any finite subset of a fooling set is also a fooling set.

(17) \( \{ 0^i 1^j : |i − j| \leq 39 \} \) is regular.

Solution: False. \( \{ 0^{i_0} : i \geq 0 \} \) is an infinite-size fooling set for the language.

(18) \( \{ 0^i 1^j : |i − j| \geq 39 \} \) is regular.

Solution: False. \( \{ 0^{i_0} : i \geq 0 \} \) is an infinite-size fooling set for the language.

(19) If \( L \) is recognized by an \( n \)-state DFA, then \( L^* \) is recognized by an \( (n + 2) \)-state NFA.


(20) If \( L \) is recognized by an \( n \)-state NFA, then \( \Sigma^* \setminus L \) is recognized by an \( n \)-state NFA.

Solution: False. Exchanging accepting and non-accepting states does not always alter the language recognized by an NFA to its complement.

Rubric: Each correct answer gets 1 point; each incorrect answer gets –0.5 points; leaving the answer blank gets 0 points. The total score can be negative.

2. Decide if the following languages are regular or not, and justify your answers. Assume that \( \Sigma = \{ 0, 1 \} \).

(a) \( \{ xwy : w, x, y \in \Sigma^+ \} \)

Solution: The language is regular. There are only finitely many strings not in the language; by the fact that any language of finite size is regular and regularity is preserved under complement, the result follows.

We claim that any string of length at least 6 is in the language. For an arbitrary string \( z \) of length at least 6, write \( z \) as \( az'b \) where both \( a \) and \( b \) are a single symbol in \( \Sigma \). Now \( z' \) has length at least 4.

• First, there are no consecutive \( 00 \) or \( 11 \) in \( z' \); otherwise we can take \( ww \) to be the pair of consecutive symbols and \( x \) (\( y \)) to be the prefix (suffix) left in \( z \), respectively.

• Now \( z' \) must be altered between \( 0s \) and \( 1s \). However, since \( z' \) has length at least 4, there must be an occurrence of \( 0101 \) or \( 1010 \). In this case we can take \( w \) to be \( 01 \) or \( 10 \) and \( x, y \) to be the rest.
3. Let \( L \) be a regular language. Define the *stutter* function on any string as follows:

\[
stutter(w) := \begin{cases} \\
\varepsilon & \text{if } w = \varepsilon; \\
aa \cdot stutter(x) & \text{if } w = a \cdot x \text{ for some } a \in \Sigma \text{ and } x \in \Sigma^*. 
\end{cases}
\]

Construct an NFA recognizing the following language.

\[
destutter(L) := \{ w \in \Sigma^* : stutter(w) \in L \}. 
\]

**Solution:** Let \( M \) be a DFA recognizing \( L \), described by the tuple \((\Sigma, Q, s, A, \delta)\). Construct an NFA \( N = (\Sigma, Q, s, A, \delta') \) recognizing \( destutter(L) \) by modifying \( M \) as follows:

- Alphabet \( \Sigma \), set of states \( Q \), starting state \( s \), accepting states \( A \) all stay the same.
- Define new transition function \( \delta' \) as

\[
\delta'(q, a) := \delta(\delta(q, a), a).
\]

We prove that \( \delta^*(q, stutter(w)) = (\delta')^*(q, w) \) for any state \( q \) and any string \( w \) by induction of the length of \( w \). Let \( w \) be an arbitrary string in \( \Sigma^* \).

- If \( w = \varepsilon \),
  \[
  \delta^*(q, stutter(w)) = \delta^*(q, \varepsilon) = q = (\delta')^*(q, w).
  \]
- If \( w = ax \) for some symbol \( a \) and string \( x \), by induction and the recursive definition of the *stutter* function,

\[
\begin{align*}
\delta^*(q, stutter(w)) &= \delta^*(q, aa \cdot stutter(x)) \\
&= \delta^*(\delta(q, a), stutter(x)) \\
&= \delta'(\delta'(q, a), stutter(x)) \\
&= (\delta')^*(\delta'(q, a), x) \\
&= (\delta')^*(q, w).
\end{align*}
\]

Now the statement is proved by induction, which implies \( w \) is accepted by \( N \) if and only if \( stutter(w) \) is accepted by \( M \). Therefore NFA \( N \) correctly recognizes \( destutter(L) \).
4. Let $N$ be an NFA with $n$ states. Assume that $N$ does not accept every string (that is, $L(N) \neq \Sigma^*$). Prove that there is a string rejected by $N$ of length at most $2^n$.

**Solution:** Construct an equivalent DFA $M$ accepting the same language as the given NFA $N$ by the subset construction. DFA $M$ has $2^n$ states.

By assumption, there is a string $w$ rejected by $N$ and thus by $M$. Because $M$ is deterministic, there is a unique walk in $M$ from the starting state $s$ to a rejecting state $q$ associated with $w$. This implies that $q$ is reachable from $s$ in $M$, which implies there is a path from $s$ to $q$ in $M$ of length at most the number of states in $M$, which is $2^n$. The concatenation of symbols associated with the transition edges on such path is another word $w'$ rejected by $M$ (and thus by $N$ as well). ■