1. For each statement below, check “True” if the statement is always true and “False” otherwise. Each correct answer is worth +1 point; each incorrect answer is worth −½ point; checking “I don’t know” is worth +¼ point; and flipping a coin is (on average) worth +¼ point.

(a) If 100 is a prime number, then Jeff is the Queen of England.

Solution: The implication \( p \rightarrow q \) is logically equivalent to \( \overline{p} \lor q \). The premise (100 is prime) is false, so the implication is true, whether or not Jeff is actually the Queen of England.

(b) The language \( \{0^m0^n0^m | m, n \geq 0\} \) is regular.

Solution: This is the language is described by the regular expression \( (00)^* \).

(c) For all languages \( L \), the language \( L^* \) is regular.

Solution: \( L^* \) is regular for every regular language \( L \) by definition, but if \( L \) is not a regular language, \( L^* \) may or may not be regular. For example, if \( L \) is the set of all balanced strings of parentheses — described by the grammar \( S \rightarrow \varepsilon \mid SS \mid (S) \) — then \( L^* \) is not regular (because \( L^* = L \)).

(d) For all languages \( L \subset \Sigma^* \), if \( L \) can be recognized by a DFA, then \( \Sigma^* \setminus L \) cannot be represented by a regular expression.

Solution: If \( L \) is regular, then \( \Sigma^* \setminus L \) is also regular.

(e) For all languages \( L \) and \( L' \), if \( L \subseteq L' \) and \( L' \) is regular, then \( L \) is regular.

Solution: Consider \( L = \{0^n1^n | n \geq 1\} \) and \( L' = \varepsilon \cup 0^*1^* \), or even better, \( L' = \Sigma^* \).
(f) For all languages $L$, if $L$ has a finite fooling set, then $L$ is regular.

Solution: A fooling set is a set of strings in which every pair of elements has a distinguishing suffix. So any subset of a fooling set is also a fooling set. In particular, the empty language is (vacuously) a fooling set for every language $L$. A language $L$ is regular if every fooling set for $L$ is finite.

(g) Let $M = (\Sigma, Q, s, A, \delta)$ and $M' = (\Sigma, Q, s, Q \setminus A, \delta)$ be arbitrary NFAs with identical alphabets, states, starting states, and transition functions, but with complementary accepting states. Then $L(M) \cup L(M') = \Sigma^*$.

Solution: Suppose $\delta(q, a) = \emptyset$ for all states $q \in Q$ and all symbols $a \in \Sigma$. Then $\delta^*(s, w) = \emptyset$ for every non-empty string $w$, and therefore $L(M) \cup L(M') = \{\epsilon\}$.

(h) Let $M = (\Sigma, Q, s, A, \delta)$ and $M' = (\Sigma, Q, s, Q \setminus A, \delta)$ be arbitrary NFAs with identical alphabets, states, starting states, and transition functions, but with complementary accepting states. Then $L(M) \cap L(M') = \emptyset$.

Solution: For any string $w \in \Sigma^*$, if the set $\delta^*(s, w)$ contains both a state in $A$ and a state in $Q \setminus A$, then both machines accept $w$.

(i) For all context-free languages $L$ and $L'$, the language $L \cdot L'$ is also context-free.

Solution: Suppose $L$ is generated by a context-free grammar with starting variable $A$, and $L'$ is generated by a context-free grammar with starting variable $B$, where the two grammars have disjoint variables. Then $L \cdot L'$ is generated by the union of the two grammars, with a new start state $S$ and a new production $S \rightarrow AB$.

(j) Every regular language is context-free.

Solution: See the lecture notes for a proof.

Rubric: +1 for each correct answer, $-\frac{1}{2}$ for each incorrect answer, $+\frac{1}{4}$ for each "I don't know." Explanations (in gray) are not required.
2. For each of the following languages over the alphabet $\Sigma = \{0, 1\}$, either prove that the language is regular or prove that the language is not regular. Exactly one of these two languages is regular.

(a) $\{w0^n w \mid w \in \Sigma^+ \text{ and } n > 0\}$

Solution: This language is not regular.

Consider the set $F = 1^+0$.

Let $x$ and $y$ be arbitrary distinct strings in $F$.

Then $x = 1^i0$ and $y = 1^j0$ for some positive integers $i \neq j$.

Let $z = 1^i$.

Then $xz = 1^i01^i \in L$, but $yz = 1^j01^i \notin L$.

So $F$ is a fooling set for $L$. Because $F$ is infinite, $L$ cannot be regular.

Rubric: 5 points = 2 for “not regular” + 1 for infinite fooling set + 2 for fooling set proof (as in HW2 problem 2). This is not the only correct solution.

(b) $\{0^n w 0^n \mid w \in \Sigma^+ \text{ and } n > 0\}$

Solution: This language is regular. Specifically, this is the language $0(0 + 1)^+0$.

For every string $x \in L$, we have $x = 0^n w 0^n$ for some $w \in \Sigma^+$ and $n > 0$; in particular, we have $x = 0y0$ for some string $y = 0^{n-1}w0^{n-1} \in \Sigma^+$. Conversely, for every non-empty string $w \in (0 + 1)^+$, the string $0w0 = 0^1w0^1$ is in $L$.

Solution: This language is regular. Here is a DFA that recognizes it:

![DFA Diagram]

Rubric (tentative): 5 points = 2 for “regular” + 3 for DFA/NFA/expression (as in HW1 problem 4). These are not the only correct answers. A proof that the given DFA, NFA, or regular expression is correct is not required. Similarly, an English description/explanation is not required, but it may help us give you partial credit.
3. Let \( L = \{ 1^n0^n \mid n \leq m \leq 2n \} \) and let \( G \) be the following context free-grammar:

\[ S \rightarrow 1S \circ | 11S \circ | \varepsilon \]

(a) **Prove** that \( L(G) \subseteq L \).

**Solution:** Let \( w \) be an arbitrary string in \( L(G) \). Assume that \( L \) contains every string \( x \in L(G) \) such that \( |x| < |w| \). There are three cases to consider, mirroring the production rules in \( G \).

- Suppose \( w = 1x\circ \) for some string \( x \in L(G) \). The induction hypothesis implies that \( x \in L \), and therefore \( x = 1^m0^n \) for some integers \( m \) and \( n \) where \( n \leq m \leq 2n \). It follows that \( w = 1^{m+1}0^{n+1} \in L \), because \( n + 1 \leq m + 1 \leq 2n + 2 \).
- Suppose \( w = 11x\circ \) for some string \( x \in L(G) \). The induction hypothesis implies that \( x \in L \), and therefore \( x = 1^m0^n \) for some integers \( m \) and \( n \) where \( n \leq m \leq 2n \). It follows that \( w = 1^{m+2}0^{n+1} \in L \), because \( n + 1 \leq m + 2 \leq 2n + 2 \).
- Finally, if \( w = \varepsilon \), then \( w = 1^00^0 \in L \), because \( 0 \leq 0 \leq 2 \cdot 0 \).

In all cases, we conclude that \( w \in L \).

**Rubric:** 5 points (standard induction rubric)

(b) **Prove** that \( L \subseteq L(G) \).

**Solution:** Let \( w \) be an arbitrary string in \( L \); by definition, \( w = 1^n0^n \) for some integers \( m \) and \( n \) where \( n \leq m \leq 2n \). Assume that \( L(G) \) contains every string \( x \in L \) such that \( |x| < |w| \). There are three cases to consider:

- If \( n = 0 \), then \( m = 0 \) and thus \( w = \varepsilon \), so the production rule \( S \rightarrow \varepsilon \) implies that \( w \in L(G) \).
- Suppose \( 0 < n = m \). Then we can write \( w = 1x\circ \), where \( x = 1^{n-1}0^{n-1} \). The inequalities \( n - 1 \leq n - 1 \leq 2n - 2 \) imply that \( x \in L \), and therefore \( x \in L(G) \) by the induction hypothesis. The production rule \( S \rightarrow 1S\circ \) gives us the derivation \( S \rightarrow 1S\circ \rightarrow 1x\circ = w \).
- Finally, suppose \( 0 < n < m \). We must have \( n \geq 1 \) and \( m \geq 2 \), so we can write \( w = 11x\circ \), where \( x = 1^{m-2}0^{n-1} \). The inequalities \( n - 1 \leq m - 2 \leq 2n - 2 \) imply that \( x \in L \), and therefore \( x \in L(G) \) by the induction hypothesis. The production rule \( S \rightarrow 11S\circ \) gives us the derivation \( S \rightarrow 11S\circ \rightarrow 11x\circ = w \).

In all cases we conclude that \( w \in L(G) \).

**Rubric:** 5 points, standard induction rubric (scaled). –1 if the case analysis is not obviously exhaustive, or for implicitly making stronger assumptions than the stated case conditions. This is not the only correct proof.
4. For any language \( L \), let \( \text{Prefixes}(L) := \{ x \mid xy \in L \text{ for some } y \in \Sigma^* \} \) be the language containing all prefixes of all strings in \( L \). For example, if \( L = \{000, 100, 111\} \), then \( \text{Prefixes}(L) = \{ \epsilon, 0, 00, 01, 11, 000, 100, 110, 111\} \).

**Prove** that for any regular language \( L \), the language \( \text{Prefixes}(L) \) is also regular.

**Solution (one new state):** Let \( M = (\Sigma, Q, s, A, \delta) \) be an arbitrary DFA that recognizes \( L \).

Without loss of generality, we assume that \( Q \) contains a unique fail state that cannot reach any accept state; that is, for every state \( q \neq \text{fail} \), there is at least one string \( w \in \Sigma^* \) such that \( \delta(q, w) \in A \). Otherwise, we can merge all states that cannot reach \( A \) to get a smaller DFA that still recognizes \( L \).

We define an NFA \( M' = (\Sigma, Q', s', A', \delta') \) as follows:

\[
Q' = Q \cup \{ \text{win} \} \\
A' = \{ \text{win} \} \\
\delta'(\text{fail}, \epsilon) = \emptyset \\
\delta'(\text{fail}, a) = \emptyset \quad \text{for all } a \in \Sigma \\
\delta'(q, \epsilon) = \{ \text{win} \} \quad \text{for all } q \neq \text{fail} \\
\delta'(q, a) = \{ \delta(q, a) \} \quad \text{for all } q \neq \text{fail} \text{ and } a \in \Sigma
\]

In other words, we add a new state \( \text{win} \) with \( \epsilon \)-transitions from every non-fail state, and then make \( \text{win} \) the only accept state.

**Solution (ghost copy):** Let \( M = (\Sigma, Q, s, A, \delta) \) be an arbitrary DFA that recognizes \( L \).

We define an NFA \( M' = (\Sigma, Q', s', A', \delta') \) as follows:

\[
Q' = Q \times \{0, 1\} \\
s' = (s, 0) \\
A' = \{(q, 1) \mid q \in Q\} \\
\delta'((q, 0), \epsilon) = \{(q, 1)\} \quad \text{for all } q \in Q \\
\delta'((q, 0), a) = \{ \delta(q, a), 0 \} \quad \text{for all } q \in Q \text{ and } a \in \Sigma \\
\delta'((q, 1), \epsilon) = \{ \delta(q, a), 0 \} \quad \text{for all } q \in Q \\
\delta'((q, 1), a) = \emptyset \quad \text{for all } q \in Q \text{ and } a \in \Sigma
\]

In other words, we make a duplicate “ghost” copy of \( Q \), replace all the transitions in the ghost copy with \( \epsilon \)-transitions, and then add an \( \epsilon \)-transition from each original state to its ghost counterpart. Only states in the second copy actually accept. The original copy of \( M \) reads the prefix \( x \) and non-deterministically passes control to the ghost copy of \( M \), which “reads” the missing suffix \( y \).
Rubric (tentative): 10 points =
+ 2 for a formal, complete, and unambiguous description of an NFA. (No credit for the rest of the problem if this is missing.)
+ 6 for a correct NFA = 3 for accepting all prefixes + 3 for accepting only prefixes, except:
  − 1 for a single typo or similar mistake
  − 1 for $\varepsilon$-transitions from fail states to accept state
  − 2 for rejecting $\varepsilon$ even when $\varepsilon \notin L$. ($\varepsilon$ is a prefix of every string.)
+ 2 for a brief English explanation. A formal proof of correctness not required.

(This rubric is similar to HW2 problems 1 and 3.) These are not the only correct solutions.
5. For each of the following languages \( L \), give a regular expression that represents \( L \) and describe a DFA that recognizes \( L \). You do not need to prove that your answers are correct.

(a) The set of all strings in \( \{0, 1\}^* \) that contain either both or neither of the substrings \( 01 \) and \( 10 \).

**Solution:** \( 0^* + 1^* + 0^+ 1^+ 0(0 + 1)^* + 1^+ 0^+ 1(0 + 1)^* \)

Any string that contains neither \( 01 \) nor \( 10 \) must be in the language \( 0^* + 1^* \). Any string that contains both \( 01 \) and \( 10 \) has either a run of \( 0 \)'s surrounded by \( 1 \)'s or a run of \( 1 \)'s surrounded by \( 0 \)'s.

![DFA Diagram]

**Rubric:** 5 points = 2½ for regular expression + 2½ for DFA (using rubric from HW1). Only one regular expression is required. Explanation (in gray) is not required. This is not the only correct solution.
(b) The set of all strings in $\{0, 1\}^*$ that do not contain the substring $1010$.

Solution: Here are two regular expressions:

- $0^*1^*(000^*11^* + 00^*111^*)^*0^*1^*$ — Any run of $0$s either starts the string, has length at least 2, is followed by a run of $1$s of length at least 2, or is the last run of $0$s.

- $(0 + 1(1 + 011)^*00)^*(\epsilon + 1(1 + 011)^*(\epsilon + 0 + 01))$ — Extracted from the DFA below using Han and Wood’s algorithm.

Rubric: 5 points = $2\frac{1}{2}$ for regular expression + $2\frac{1}{2}$ for DFA (using rubric from HW1). Only one regular expression is required. Explanation (in gray) is not required. These are not the only correct solutions.