I have skipped drawing pictures to make an all-text answer key that emphasizes the ideas and strategies.

(1) (21 pts.)

Let \( A = \{ x \in \{a, b\}^* : \text{bb is a substring of } x \text{ and } \#a(x) \text{ is even} \} \). Design a deterministic finite automaton (DFA) \( M \) such that \( L(M) = A \). A node-arc diagram that shows the start and final states clearly is good enough—you need not write out tables or “\( M = (Q, \Sigma, \delta, s, F) \ldots \)”

Answer: Make a grid with two rows standing for \( \#a(x) \text{ even} \) and \( \#a(x) \text{ odd} \), and with three columns labeled 0,1,2 standing for the “progress” toward the substring \( \text{bb} \). Arcs on \( b \) always go straight across to the right and loop in the rightmost column. Arcs on \( a \) go between the rows vertically in columns 0 and 2, but—this was the main tricky point—from column 1 they go diagonally back to column 0 because getting an \( a \) after a \( b \) destroys all the progress toward \( \text{bb} \). The start state is in the first row and column since 0 is an even number; the unique accepting state is in row “even” and the rightmost column. The row/column labels suffice as comments. [The Cartesian product construction for intersection—of a 2-state DFA for “\( \#a \) even” and a 3-state DFA for “substring \( \text{bb} \)”—gives the same answer.]

(2) (18 + 18 = 36 pts.)

Let \( N \) be the NFA defined by \( N = (Q, \Sigma, \delta, s, F) \) with \( Q = \{1, 2, 3\} \), \( \Sigma = \{a, b\} \), \( s = 1 \), \( F = \{2\} \), and \( \delta \) given by the arcs \((1, b, 2)\), \((1, \epsilon, 2)\), \((1, a, 3)\), \((2, b, 3)\), \((3, b, 1)\), and \((3, a, 2)\) (diagram omitted in this key).

(a) Calculate a DFA \( M \) such that \( L(M) = L(N) \).

(b) Calculate a regular expression \( r \) such that \( L(N) = L(r) \).

Answer: (a) The key point is that the \( \epsilon \)-arc means “Whenever 1, then also 2” and makes the start state \( S = \{1, 2\} \) for the DFA, not just \( \{1\} \). It also makes the subset states \( \{1\} \) and \( \{1, 3\} \) impossible, leaving a max of 6 possible states. In fact, the DFA maxes out on those states:

\[
\Delta(S, a) = \delta(1, a) \cup \delta(2, a) = \{3\} \cup \emptyset = \{3\}; \\
\Delta(S, b) = \delta(1, b) \cup \delta(2, b) = \{2\} \cup \{3\} = \{2, 3\}; \\
\Delta(\{3\}, a) = \delta(3, a) = \{2\}; \\
\Delta(\{3\}, b) = \delta(3, b) = \{1, 2\}; \quad (\text{not just } \{1\}) \\
\Delta(\{2, 3\}, a) = \delta(2, a) \cup \delta(3, a) = \{2\}; \\
\Delta(\{2, 3\}, b) = \delta(2, b) \cup \delta(3, b) = \{3\} \cup \{1, 2\} = \{1, 2, 3\}; \\
\Delta(\{2\}, a) = \delta(2, a) = \emptyset; \quad (!) \\
\Delta(\{2\}, b) = \delta(2, b) = \{3\}; \\
\Delta(\{1, 2, 3\}, a) = \{2, 3\}; \\
\Delta(\{1, 2, 3\}, b) = \{1, 2, 3\};
\]
The sixth state is the dead state ∅ which goes to itself on all characters. The set of final states is every subset that includes 3, i.e., all but {3} and the dead state.

(b) The key point here is that it is easiest to work with the NFA, which in fact is already labeled nicely with state 2 the only accepting state. Hence there is no need to do any of the text’s preambles—we just eliminate state 3 and apply the 2-state formula. Now state 3 is a little bit of a bear since it has “incoming” from both state 1 and state 2 and “outgoing” arcs to both of those states. So in terms of the original regular-expression matrix

\[
T = \begin{bmatrix}
∅ & ε + b & a \\
∅ & ∅ & b \\
b & a & ∅
\end{bmatrix},
\]

we have to update all of \(T(1,1), T(1,2), T(2,1), T(2,2)\). Applying the formula for edge-label updates “bypassing” state 3 gives

\[
T = \begin{bmatrix}
ab & ε + b + aa \\
bb & ba
\end{bmatrix}
\]

The formula for \(L(N)\) is then

\[
L_{1,2} = (T(1,1) + T(1,2)T(2,2)^*T(2,1))^*T(1,2)T(2,2)^*
= (ab + (ε + b + aa)(ba)^*bb)^*(ε + b + aa)(ba)^*.
\]

There were other correct answers. Having a strategy instead of “hacking” was the most important consideration in case of any detected error.

(b) The key point here is that it is easiest to work with the NFA, which in fact is already labeled nicely with state 2 the only accepting state. Hence there is no need to do any of the text’s preambles—we just eliminate state 3 and apply the 2-state formula. Now state 3 is a little bit of a bear since it has “incoming” from both state 1 and state 2 and “outgoing” arcs to both of those states. So in terms of the original regular-expression matrix

\[
T = \begin{bmatrix}
∅ & ε + b & a \\
∅ & ∅ & b \\
b & a & ∅
\end{bmatrix},
\]

we have to update all of \(T(1,1), T(1,2), T(2,1), T(2,2)\). Applying the formula for edge-label updates “bypassing” state 3 gives

\[
T = \begin{bmatrix}
ab & ε + b + aa \\
bb & ba
\end{bmatrix}
\]

The formula for \(L(N)\) is then

\[
L_{1,2} = (T(1,1) + T(1,2)T(2,2)^*T(2,1))^*T(1,2)T(2,2)^*
= (ab + (ε + b + aa)(ba)^*bb)^*(ε + b + aa)(ba)^*.
\]

There were other correct answers. Having a strategy instead of “hacking” was the most important consideration in case of any detected error.

(3) (5 × 3 = 15 pts.) True/False, no justifications needed.

(a) If \(A^* = A\), then the language \(A\) includes the empty string.

(b) If \(A\) and \(B\) are regular languages recognized by 3-state DFAs, then \(A \cap B\) can be recognized by a 6-state DFA.

(c) The empty relation on a nonempty set is transitive.

(d) The intersection of two non-regular languages is always non-regular.

(e) If there is a string \(w\) such that no string \(x\) in a regular language \(A\) has \(w\) as a substring, then every DFA \(M\) such that \(L(M) = A\) has a dead state.

(a) True, since \(A^*\) always includes the empty string.

(b) False—it can require \(3 \times 3 = 9\) states, not \(3 + 3 = 6\). An example where it does is when \(A\) is the language of strings \(x\) for which \(#a(x)\) is a multiple of 3, and \(B\) ditto for \(#b(x)\).

(c) True—although the opening “for all \(x, y, z\)” part of the definition of transitivity does not “default” since the domain is nonempty, the next part is “if \(xRy\) and...” which does default, since \(R\) is empty so \(xRy\) is never true. So by the default rule for an if-then clause whose “if” part is false, it gives true.
(d) False—because we could intersect the non-regular language \{a^n b^n\} with its equally non-regular complement and get $\emptyset$, which is a regular language.

(e) True—Because a prefix is a case of having a substring, no word beginning with $w$ belongs to $L(M)$. Hence $M$ must process $w$ from start to the dead state (or to one of a cluster of states that are all dead).

(4) (7 pts.)
Let $L = \{x \in \Sigma^* : \text{both of the substrings } aa \text{ and } bb \text{ occur in } x, \text{ each at least once}\}$. Write a regular expression $r$ such that $L(r) = L$. (Hint: Break into two cases, one for $aa$ coming first, the other for $bb$ coming first.)

Answer: The suggested breakdown makes $r = r_a + r_b$ where $r_a = (a+b)^*aa(a+b)^*bb(a+b)^*$ and $r_b = (a+b)^*bb(a+b)^*aa(a+b)^*$.

(5) (21 pts.)
Define $L = \{x \in \{a,b\}^* : |x| \text{ is odd and the middle character of } x \text{ is a } \text{'b'}\}$. For instance, $b$, $aba$, and $baabaaa$ belong to $L$, but $\epsilon$, $bbbb$, and $bbabb$ do not.

Prove using the Myhill-Nerode technique that $L$ is not a regular language. Hint: take $S = (aa)^*$ or $S = (bb)^*$.

Answer: The hint was not necessary but it avoids any fuss about whether the total length is odd or even: Take $S = (aa)^*$. Clearly $S$ is infinite. Let any $x, y \in S$, $x \neq y$, be given. Then there are distinct numbers $m, n \geq 0$ such that $x = (aa)^m$ and $y = (aa)^n$. Take $z = b(aa)^m$. Then $xz = (aa)^m b(aa)^m \in L$, but $yz = (aa)^n b(aa)^m \notin L$ because with $m \neq n$ the lone $b$ cannot be in the middle. Since $x, y \in S$ are arbitrary, $S$ is PD for $L$, and since $S$ is infinite, $L$ is non-regular by the Myhill-Nerode Theorem.

Pretty much the same argument works with $S = a^*$ because then $xz = a^m ba^n$ still has odd length, while in $yz = a^m ba^n$ you can just ignore the possibility that $m + 1 + n$ is even—it gives $yz \notin L$ anyway. The arguments can also work with $S = (bb)^*$, but then it is important to say “without loss of generality $m < n$” after letting $x = (bb)^m$ and $y = (bb)^n$. Then the choice $z = a(bb)^m$ puts $xz \notin L$ but $yz \in L$ because $|yz| = 2n + 1 + 2m$ is definitely odd and one of the first $n$ $b$’s will be in the middle thanks to $n > m$.

End of Exam.