(1) Convert the following finite automata $N_1$ and $N_2$ into regular expressions $r_1$ and $r_2$ such that $L(r_1) = L(N_1)$ and $L(r_2) = L(N_2)$. No comments are needed on the final products, but you should show your work in the conversion process clearly—in particular noting any “reasonable shortcuts.” (2 × 18 = 36 pts.)

$N_1 = (Q_1, \Sigma, \delta_1, s_1, F_1)$ with $Q_1 = \{1, 2, 3, 4\}$, $\Sigma = \{a, b\}$, $s_1 = 1$, $F_1 = \{1, 2\}$, and $\delta_1 = \{(1, a, 2), (1, b, 3), (2, a, 3), (2, b, 1), (3, a, 4), (3, b, 1), (4, a, 3)\}$.

$N_2 = (Q_2, \Sigma, \delta_2, s_2, F_2)$ with $Q_2 = \{1, 2, 3, 4\}$, $\Sigma = \{a, b\}$, $s_2 = 1$, $F_2 = \{2, 4\}$, and $\delta_2 = \{(1, a, 2), (1, b, 4), (2, a, 3), (2, b, 4), (3, a, 1), (3, b, 3), (4, a, 1), (4, b, 3), (4, \epsilon, 2)\}$. Note the last $\epsilon$-arc.

**Answer:** $N_1$ has just one final state besides its start state, so we can dispense with the text step of adding a new final state (and a new start state). Eliminate the non-accepting states 4 and 3 will get us down to the 2-state base case where we have $L(N_1) = L_{1,1} \cup L_{1,2}$ and can use the formulas in lecture. State 4 can be pinched right off: replace the NFA instructions $T(3, a, 4)$ and $T(4, a, 3)$ by the single GNFA arc $(3, aa, 3)$. The matrix of the resulting 3-state GNFA is:

$$T = \begin{bmatrix} \emptyset & a & b \\ b & \emptyset & a \\ b & \emptyset & aa \end{bmatrix}$$

To eliminate 3, we have incoming from 1 on $b$ and 2 on $a$, and outgoing just to 1 on $b$, so we update $T(1,1)$ and $T(2,1)$:

- $T(1,1)_{\text{new}} = T(1,1)_{\text{old}} \cup T(1,3)T(3,3)^*T(3,1) = \emptyset \cup b(aa)^*b = b(aa)^*b$
- $T(2,1)_{\text{new}} = T(2,1)_{\text{old}} \cup T(2,3)T(3,3)^*T(3,1)$
  - $= b \cup a(aa)^*b$.

If you wrote $T(1, 1) = \epsilon$ and got $T(1, 1)_{\text{new}} = \epsilon + b(aa)^*b$ instead it won’t matter, because the new $T(1,1)$ will wind up inside a $\ast$ where the identity $(\emptyset \cup R)^* = (\epsilon \cup R)^* = R^*$ will apply for whatever rest of the regular expression $R$ you get. But you can’t leave out the initial $b$ in $T(2, 1)$ (and wouldn’t be able to leave out $\epsilon$ if it were $\epsilon$). There was no change to $T(2, 1)$ or $T(2, 2)$. The new “$T$-matrix” is:

$$T' = \begin{bmatrix} b(aa)^*b & a \\ b \cup a(aa)^*b & \emptyset \end{bmatrix}$$

The fact of having the “Gamma” part be $\emptyset$ (or $\epsilon$, doesn’t matter since it’s a self-loop) makes the formulas easier. Compared to the general formulas, $T'(2, 2)^* = \epsilon$ so it drops out, leaving:

$$
L_{1,1} = (T'(1, 1) \cup T'(1, 2)T'(2, 1))^* = (b(aa)^*b + a(b + a(aa)^*b))^*
$$

$$
L_{1,2} = L_{1,1}T'(1, 2) = L_{1,1}a
$$

$$
L(N_1) = L_{1,1} \cup L_{1,2} = L_{1,1}(\epsilon \cup T'(1, 2))
= (b(aa)^*b + a(b + a(aa)^*))^*(\epsilon + a).
$$
There are other equivalent forms; in order to grade them efficiently it is important to be able to follow the strategy.

$N_2$ has two accepting states different from the start state, which is usually the trigger to follow the text—and there was nothing wrong with doing so. However, the $\epsilon$-arc from 4 to 2 means that we can “de-$F$” state 4 without changing the language or processing of the machine. So we can eliminate state 4 after all, and then 3, without having to mess with an extra state and an extra elimination step. From $\delta_2 = \{(1, a, 2), (1, b, 4), (2, a, 3), (2, b, 4), (3, a, 1), (3, b, 3), (4, a, 1), (4, b, 3), (4, \epsilon, 2)\}$ we see 4 has incoming from 1 and 2 and outgoing to 1 and 3 and 2—we can’t ditch the $\epsilon$-arc. This isn’t as bad as $3 \times 3$ but still means $2 \times 3 = 6$ updates, yuck. Let’s try eliminating state 3 first instead. It has incoming from 2 and 4 and outgoing to 1 only—$(3, b, 3)$ is a self-loop. Yay—only 2 updates:

$$
T(2, 1)_{\text{new}} = T(2, 1)_{\text{old}} + T(2, 3)T(3, 3)^*T(3, 1) = ab^*a
$$
$$
T(4, 1)_{\text{new}} = T(4, 1)_{\text{old}} + T(4, 3)T(3, 3)^*T(3, 1) = a + bb^*a.
$$

The new $T$-matrix now has state 4 in its 3rd column and row:

$$
\begin{bmatrix}
\emptyset & a & b \\
ab^*a & \emptyset & b \\
a + bb^*a & \epsilon & \emptyset
\end{bmatrix}
$$

Again, the $\epsilon$ is important because $T(4, 2)$ (old and new) is not a self-loop, and indeed it gives us ‘outgoing’ from 4 to 2 as well as 1. And we have ‘incoming’ from both 1 and 2 on $b$ so we get the full $2 \times 2$ update need, but that’s still better than the $2 \times 3 = 6$ we were facing before. Rolling up the sleeves, and skipping the “old” and “new” subscripts since we know how to execute lines of code like $x = x + 3$, we get:

$$
T(1, 1) = T(1, 1) + T(1, 4)T(4, 4)^*T(4, 1) = b(a + bb^*a)
$$
$$
T(1, 2) = T(1, 2) + T(1, 4)T(4, 2)^*T(4, 1) = a + b
$$
$$
T(2, 1) = T(2, 1) + T(2, 4)T(4, 4)^*T(4, 1) = ab^*a + b(a + bb^*a)
$$
$$
T(2, 2) = T(2, 2) + T(2, 4)T(4, 4)^*T(4, 2) = b.
$$

From this we get $L(N_2) = L_{1,2} = (T(1, 1) + T(1, 2)T(2, 2)^*T(2, 1))T(1, 2)T(2, 2)^*$. We could use the other formula for $L_{1,2}$ in lecture notes but that one stars $T(1, 1)$ twice which makes it longer. You can leave it at this, but just to see the substitution:

$$
L_{1,2} = (b(a + bb^*a) + (a + b)b^*(ab^*a + b(a + bb^*a)))^*(a + b)b^*.
$$

(2) Given any strings $u$ and $x$ (with $u \neq \epsilon$), define $\#u(x)$ to be the number of times $u$ occurs as a substring of $x$. For example, if $u = 010$ and $x = 01010$ then $\#u(x) = 2$ because two occurrences are counted even though they overlap. Using the Myhill-Nerode technique (not the Pumping Lemma), prove that two the following languages are not regular—and give a regular expression for the one that is regular. ($3 \times 12 = 36$ pts.)
(a) \( L_1 = \{ x \in \{0, 1\}^* : \#00(x) > \#11(x) \} \).

(b) \( L_2 = \{ x \in \{0, 1\}^* : \#01(x) > \#10(x) \} \).

(c) \( L_3 = \{ x0y : \#0(x) = \#1(y) \} \).

Answer: \( L_1 \) is not regular. Take \( S = (00)^* \). Clearly \( S \) is infinite. Let any \( x, y \) with \( x \neq y \) be given. Then there are numbers \( m, n \geq 0 \) with \( m < n \) such that wlog. we can write \( x = (00)^m \) and \( y = (00)^n \). Take \( z = (11)^m \). Then \( xz = (00)^m(11)^m \notin L_1 \) because there are not more 00s than 11s, but \( yz = (00)^n(11)^m \) is in \( L_1 \) since \( n > m \). Thus \( S \) is PD for \( L_1 \), and since \( S \) is infinite, \( L_1 \) is non-regular by the Myhill-Nerode theorem.

A literal cut-and-paste of the above proof for \( L_1 \), substituting 01 for 00 and 10 for 11, would go: “Take \( S = (01)^* \). Clearly \( S \) is infinite. Let any \( x, y \) with \( x \neq y \) be given. Then there are numbers \( m, n \geq 0 \) with \( m < n \) such that wlog. we can write \( x = (01)^m \) and \( y = (01)^n \). Take \( z = (10)^m \). Then \( xz = (01)^m(10)^m \notin L_2 \) because there are not more 01s than 10s, but \( yz = (01)^n(10)^m \) is in \( L_2 \) since \( n > m \)…” There, however, it breaks down: Consider \( m = 1 \) and \( n = 3 \). We get \( yz = 01010110 \). We want to think there is just one ‘10’ in \( yz \) but look: there are two more occurrences in the “\( y \)” part. So \( \#01(yz) = \#10(yz) = 3 \) which means \( yz \notin L_2 \). The “proof” goes “poof!”

In fact, occurrences of 01 and 10 must alternate because it means the whole string changing from having 0s to having 1s then back to 0s and so on. The only way we get more occurrence(s) of 01 than 10 is when the string begins with 0 and ends with 1. This condition is captured by the regular expression \( 0(0 \cup 1)^*1 \).

\( L_3 \) is not regular: Take \( S = 0^* \), clearly infinite. Let any distinct \( x, y \in S \); then wlog. we have \( x = 0^m \) and \( y = 0^n \) where \( m < n \). Take \( z = 1^m \). Now the first important part is that \( yz \) does belong to \( L_3 \) because it can be broken as \( 0^m \cdot 0 \cdot 0^{n-m-1}1^m \). This is possible since \( n > m \) so \( n - m - 1 \geq 0 \). Whereas, \( xz = 0^m1^m \) does not allow such parsing because the extra 0 in the middle leaves at most \( m - 1 \) 0’s for the “\( x \)” part to go against \( m \) 1’s for the “\( y \)” part in “\( x0y \).” So \( xz \notin L_3 \), so \( L_3(xz) \neq L_3(yz) \), so \( S \) is infinite and PD for \( L_3 \), so \( L_3 \) is not regular.