(1) Prove that two of the following three languages are non-regular, via a Myhill-Nerode argument. For the regular one, give a regular expression. Here \( \#a(x) \) denotes the number of occurrences of the character \( a \) in the string \( x \), and more generally, \( \#w(x) \) denotes the number of occurrences of the substring \( w \) in \( x \). For example, \( \#0010(00100100) = 2 \) even though the two occurrences of the substring 0010 overlap each other. Also, for two strings \( x,y \) of the same length, \( x \oplus y \) denotes the bitwise exclusive-OR, e.g. \( 1011 \oplus 0010 = 1001 \). All three languages are over the alphabet \( \Sigma = \{0,1\} \).

(i) \( L_1 = \{ x : \#0(x) > \#1(x) \} \).

(ii) \( L_2 = \{ x : \#01(x) > \#10(x) \} \).

(iii) \( L_3 = \{ xy : |x| = |y| \land x \oplus y = 1^{|x|} \} \).

**Answer:** \( L_2 \) is regular: Every binary string has a 01 substring when it goes from 0s to 1s and then has a 10 substring when it alternates back. If it begins with 10, there will never be a stage where the number of occurrences of 01 beats that of 10. So the string must begin with 0 and end with 1: 0(0 + 1)*1. For the other languages, here are proofs by MNT:

(i) Take \( S = 0^* \). Clearly \( S \) is infinite. Let any \( x,y \in S, x \neq y \) be given. Then there are numbers \( m,n \in \mathbb{N} \) such that \( x = 0^m \) and \( y = 0^n \), where wlog. \( m < n \). Take \( z = 1^m \). Then \( xz = 0^m1^m \notin L_1 \) since \( m \) is not greater than itself, but \( yz = 0^n1^m \in L_1 \) since \( n > m \) by the “wlog.” clause. So \( L_1(xz) \neq L_1(yz) \), and since \( x,y \in S \) are arbitrary, \( S \) is PD for \( L_1 \). Since \( S \) is also infinite, \( L_1 \) is not regular by the Myhill-Nerode Theorem.

(ii) Take \( S = 0^+ \). We are avoiding the annoying “edge case” of whether the empty string belongs to \( L_3 \), though the answer is objectively yes and taking \( S = 0^* \) would be fine. But \( S = 0^+ \) is infinite and “more parsimonious.” Let any \( x,y \in S, x \neq y \) be given. Then there are numbers \( m,n \in \mathbb{N} \) such that \( x = 0^m \) and \( y = 0^n \), and for greater clarity we’ll again invoke the \( m < n \) clause though we don’t really need it. Take \( z = 1^m \). Then \( x \oplus z = 1^m \), so \( xz \in L_3 \). But with \( yz \), either it’s an odd-length string so we can’t break it in half, or when we do there will be two matched-up 0s because \( n > m \) means \( yz \) has more 0s than 1s. So \( yz \notin L_3 \), giving \( L_3(xz) \neq L_3(yz) \), which makes \( S \) PD for \( L_3 \) and proves that \( L_3 \) is not regular.

**Footnotes:** We could do (iii) almost as easily by arguing that a string of the form 0^i,1^j can be broken as needed for \( L_3 \) only if \( i = j \), since \( i < j \) gives an excess of paired-up entries that are 1. But thinking of just 0s was a convenience. That (ii) is regular warns against false “poofs” like this: “Take \( S = (01)^* \). Clearly \( S \) is infinite. Let any \( x,y \in S, x \neq y \) be given. Then there are \( m,n \in \mathbb{N} \) such that \( x = (01)^m \) and \( y = (01)^n \) where wlog. \( m < n \). Take \( z = (10)^m \). Then \( xz = (01)^m(10)^m \notin L_2 \) [which is true] but \( yz = (01)^n(10)^m \in L_2 \) since \( n > m \). Thus…” Despite the optical appearance, the last assertion is wrong because the substrings 10 and 01 occur “between the cracks” as well as what’s shown.

(2) Now consider \( L_4 = \{ x : \#010(x) = 0 \land \#101(x) = 0 \} \). Use the Myhill-Nerode technique to show that any DFA \( M \) such that \( L(M) = L_4 \) requires at least 6 states. Then design such a DFA
$M$—ideally showing how your proof guided you to it (or vice-versa). Finally explain why you can basically "collapse" $M$ into a generalized NFA with only 2 states $s, f$ such that

$$L(M) = L_{s,s} \cup L_{s,f} \cup L_{f,s} \cup L_{f,f},$$

and use that to give a regular expression for $L_4$. (12 + 6 + 9 = 27 pts., for 63 total on the set)

**Answer:** All strings of length 2 or less belong to $L_4$. The shortest strings that don’t are 010 and 101, and any string beginning with those is dead. Now we can consider pairs $x, y$ of strings and suffixes $z$ that make $L_4(xz) \neq L_4(yz)$:

- $x = \lambda, y = 0$: $z = 10$.
- $x = \lambda, y = 1$: $z = 01$.
- $x = 0, y = 1$: $z = 10$. Hence these three strings $x, y$ must all go to different states.
- $x = 0, y = 00$: “Oopsie,” no $z$ makes $L_4(0z) \neq L_4(00z)$. So those two strings can be processed from $s$ to the same state.
- $x = 0, y = 01$: Separated by $z = 0$. Can we separate 01 from others?
- $x = \lambda, y = 01$: Take $z = 0$.
- $x = 1, y = 01$: Take $z = 0$ again. It’s not needing to be separated by a different $z$ that matters, but the fact of having some separating $z$ for every pair of two from $\lambda, 0, 1, 01$. So we know we need 4 states—actually we have 5 since none of these strings is dead and 010 is.
- $x = 10$: That this is distinguished from the three strings $\lambda, 0, 1$ comes from a similar argument by symmetry. So we need only stack it up against $y = 01$. They are distinguished by $z = 0$ (or $z = 1$ etc.), so we have our 6.

So we design the DFA with states $s, q_0, q_1, q_01, q_{10}$ and $q_{010} = a$ dead state $d$. The transitions $(s, 0, q_0), (s, 1, q_1), (q_0, 1, q_{01}), (q_1, 0, q_{10})$, and $(q_{01}, 0, d), (q_{10}, 1, d)$ plus the self-loops at $d$ are automatic from the labels. We can also do $(q_0, 0, q_0)$ and $(q_1, 1, q_1)$ as self-loops. So we just need to finish $(q_{01}, 1, ?)$ and $(q_{10}, 0, ?)$ and they can go back to $q_1$ and $q_0$, respectively. Every state other than $d$ is accepting.

Now to find a regular expression for $L_4$, we can start by deleting the dead state, since it doesn’t help any processing and a (G)NFA can do without it. Then if we take a 1 out of $q_0$ to $q_{01}$, we have to go back on a 1 to $q_1$. Hence we can bypass $q_{01}$ simply by making $(q_0, 11, q_1)$. But we can’t forget about $q_{01}$ entirely because it is an accepting state—so we have to remember things that can go there by a final 1. And $(q_1, 00, q_0)$ similarly cuts $q_{10}$ out of the picture, but we have to remember to re-include a training 0. Now we’re left with 3 states, but $s$ goes out and doesn’t come back. So what we’re left with is a two-state GNFA where we can start up in either state and we have arcs

$$(q_0, 0, q_0), (q_0, 11, q_1), (q_1, 00, q_0), (q_1, 1, q_1).$$

The language we want—remembering the trailers—is

$$L_{0,0}(\lambda + 1) \cup L_{1,0}(\lambda + 1) \cup L_{0,1}(\lambda + 0) \cup L_{1,1}(\lambda + 0).$$

Now $L_{0,0} = (0 + 111*00)^*$, $L_{0,1} = L_{0,0}111^*$, $L_{1,1} = (1 + 000^*11)^*$, and $L_{1,0} = L_{1,1}000^*$. Plugging those in gives a yucky but correct regular expression for $L_4$. 