1. **Random Trivia** For each of the following questions, give a brief concise, answer. If it asks for a proofs, you don't have to do a formal research paper proof, but there should be a clear sequence logic.

   (a) For what languages (L) is L^* finite?

   Solution: L^* is finite if and only if $L = \{\epsilon\}$ or $L = \emptyset$. ∎

   (b) Describe two languages, A and B, such that $|A \cdot B| < |A| \cdot |B|$

   Solution: Let $A = \{\epsilon, a\}$ and $B = \{\epsilon, a\}$. Here, $|A \cdot B| = 3$ (the set $\{\epsilon, aa, a\}$) while $|A| \cdot |B| = 4$. Thus, $|A \cdot B| = 3 < 4 = |A| \cdot |B|$. ∎

   (c) Prove $(A \cup B)^* = (A^* \cdot B^*)^*$ for all languages A and B.

   Solution: Neglecting the outer Kleene star gives $(A \cup B) = (A^* \cdot B^*)$, for the left we have A or B, whereas for the right by setting Kleene stars to have value of (0,1) or (1,0), we have $\left(A^0 \cdot B^1\right)$ or $\left(A^1 \cdot B^0\right)$ which is also A or B. ∎

2. **Recursive definitions** Give the recursive defintion fo rthe followign languages:

   (a) $L_{2a} = \left\{w \in \{0,1\}^* \,|w \text{ has } 00 \text{ as a substring}\right\}$

   Solution: Use base case to enforce $00 \in L_a$

   - **Base case:** $00 \in L_{2a}$
   - $0x \in L_{2a}$ for $x \in L_{2a}$
   - $1x \in L_{2a}$ for $x \in L_{2a}$
   - $x0 \in L_{2a}$ for $x \in L_{2a}$
   - $x1 \in L_{2a}$ for $x \in L_{2a}$

   ■

   (b) $L_{2b} = \left\{w \in \{0,1\}^* \,|w \text{ all strings with alternating } 0\text{'s and } 1\text{'s}\right\}$

   Solution: Define two languages, L_{2b1} and L_{2b0} using a mutually recursive definition. L_{2b1} contains all strings in L_{2b} that start with 1, and L_{2b0} contains all strings in L_{2b} that start with 0.

   - **Base case:** $\varepsilon \in L_{2b0}$ and $\varepsilon \in L_{2b1}$
   - $\varepsilon \in L_{2b0}$
   - $\varepsilon \in L_{2b1}$
   - $1x \in L_{2b1}$ for $x \in L_{2b0}$
   - $0x \in L_{2b0}$ for $x \in L_{2b1}$

   Then $L_{2b} = L_{2b1} \cup L_{2b0}$ ■

   (c) $L_{2c} = \left\{w \in \{0,1\}^* \,|w \text{ has an equal number of } 0\text{'s and } 1\text{'s}\right\}$

   Solution: The language L_{2c} contains strings with an equal number of 0's and 1's. Therefore, we can arbitrarily pick two strings and concatenate them together, and the resulting string will also have an equal number of 0's and 1's.

   - **Base case:** $\varepsilon \in L_{2c}$
   - $1x0 \in L_{2c}$ for $x \in L_{2c}$
   - $0x1 \in L_{2c}$ for $x \in L_{2c}$
   - $xy \in L_{2c}$ for $x, y \in L_{2c}$

   ■

3. **Total equivalence** Prove that each of the following regular expressions is equivalent to $(0+1)^*$. You don't have to do a formal research paper proof, but there should be a clear sequence logic.

   (a) $\varepsilon + 0(0+1)^* + 1(0+1)^*$

   Solution: Note that every string except for ε starts with either the symbol 0 or 1. The term $0(0+1)^*$ covers any string that starts with 0, as it is simply the symbol 0 followed by the set of all strings. Similarly $1(0+1)^*$ covers any string that starts with 1. Therefore the union of the two terms and ε represents the set of all strings. ■

   (b) $0^* + 0^*1(0+1)^*$

   Solution: Every string either contains the symbol 1 or doesn't. The term 0^* covers all strings that does not contain 1, including ε. The second term $0^*1(0+1)^*$ covers any string that contains at least one 1, since 0^* covers any number of leading 0s and $1(0+1)^*$ is the set of all strings starting with 1. Therefore the given regular expression represents the set of all strings. ■

   (c) $\big((\varepsilon+0)(\varepsilon+1)\big)^*$

   Solution: The term $\varepsilon+0$ represents the set $\{\varepsilon, 0\}$, and $\varepsilon+1$ represents $\{\varepsilon, 1\}$. The concatenation of the two sets is $L = \{\varepsilon, 0, 1, 01\}$. Since L contains the alphabet $\Sigma = \{0, 1\}$ as a subset, the repetition L^* is equivalent to Σ^*, which is the set of all strings. ■

   (d) $(1^*0)^*(0^*1)^*$

   Solution: The term (1^*0) represents the set $L = \{\varepsilon, 0, 10, 110, 1110, \ldots\}$. The repetition L^* represents the set of all strings that ends with 0, while including ε as the only exception. Similarly, $(0^*1)^*$ represents the set of all strings that ends with 1. The concatenation of the two sets $(1^*0)^*(0^*1)^*$ includes the union $(1^*0)^* + (0^*1)^*$, since repeating (1^*0) for 0 times leaves $(0^*1)^*$, and repeating $(0^*1)^*$ for 0 times leaves (1^*0). Therefore, $(1^*0)^*(0^*1)^*$ represents the set of all strings. ■

4. **regular expressions** for each of the following languages ($\Sigma = \{0, 1\}$), give the regular expression that represents that language. You must concisely justify why you regular expression is correct (do not use finite automata).

   (a) L_{4a} = All strings except 010

   Solution: This one we can simply brute force (and is from the lab!). So it would look something like:

   $$\varepsilon + 0 + 1 + (0+1)^2 + 000 + 001 + 011 + 100 + 101 + 110 + 111 + (0+1)^4 (0+1)^*$$

   ■

   (b) L_{4b} = Strings that contain the subsequence 010

   Solution: We need to ensure that the strings contain the characters 0 - 1 - 0 in that order. Easiest way to write this is:

   $$(0+1)^* \, 0 \, (0+1)^* \, 1 \, (0+1)^* \, 0 \, (0+1)^*$$

   ■

   (c) L_{4c} = Strings that **do not** contain the subsequence 010

   Solution: The best thing to do this is try to rephrase the problem. In this case, we want all strings where if we have had 0's and then 1's already, then we cannot have another 0. We can also have as many 1's in the beginning before we put in any 0's. So we can write the regular language as:

   $$1^* 0^* 1^*$$

   ■

   (d) L_{4d} = Strings in which every occurrence of the substring 00 appears before every occurrence of the substring 11

   Solution: This problem is really saying, you can have as many runs of 0's as you want, until you do a run of 1's, then only single 0's. So we can write this language like so:

   $$0^* \left(10^+\right)^* 1^* \left(01^+\right)^*$$

   ■

   Solution: There is an alternate interpretation of this problem where every instance of 11 has to have a 00 before it. This is actually a much harder interpretation of the problem and we'll accept this interpreation of the problem as long as you go tit correct.

   The issue is how do we get it correct? So presumably in this interpretation of the problem you can't have 111 because then the right substring would not have a 00 in front of it. So you can only have a single 1 or with how ever many 0's in front of it, or 11 with atleast two 0's in front of it. So the regular expression might look somethign like this:

$$\left(0^* + \left(0^+1\right)^* + \left(00^+11\right)^*\right)^*$$

■