1. Structural Induction

(a) Consider the following recursive definition of strings Σ^* over an alphabet Σ.
Basis Step: ϵ is a string
Recursive Step: If w is a string and $a \in \Sigma$ is a character then wa is a string.
Recall the following recursive definition of the function len:

$$\begin{aligned} \mathsf{len}(\epsilon) &= 0 \\ \mathsf{len}(wa) &= 1 + \mathsf{len}(w) \end{aligned}$$

Now, consider the following recursive definition:

$$\begin{aligned} \mathsf{double}(\epsilon) &= \epsilon \\ \mathsf{double}(wa) &= \mathsf{double}(w)aa. \end{aligned}$$

Prove that for any string x, $\mathsf{len}(\mathsf{double}(x)) = 2\mathsf{len}(x)$.

Solution:

For a string x, let $\mathsf{P}(x)$ be "$\mathsf{len}(\mathsf{double}(x)) = 2\mathsf{len}(x)$. We prove $\mathsf{P}(x)$ for all strings $x \in \Sigma^*$ by structural induction.

Base Case. We show $\mathsf{P}(\epsilon)$ holds. By definition $\mathsf{len}(\mathsf{double}(\epsilon)) = \mathsf{len}(\epsilon) = 0$. On the other hand, $2\mathsf{len}(\epsilon) = 0$ as desired.

Induction Hypothesis. Suppose $\mathsf{P}(w)$ holds for some arbitrary string $w \in \Sigma^*$.

Induction Step. We show that $\mathsf{P}(wa)$ holds for any character $a \in \Sigma$.

$$\begin{aligned} \mathsf{len}(\mathsf{double}(wa)) &= \mathsf{len}(\mathsf{double}(w)aa) && \text{[By Definition of } \mathsf{double}] \\ &= 1 + \mathsf{len}(\mathsf{double}(w)a) && \text{[By Definition of } \mathsf{len}] \\ &= 1 + 1 + \mathsf{len}(\mathsf{double}(w)) && \text{[By Definition of } \mathsf{len}] \\ &= 2 + 2\mathsf{len}(w) && \text{[By IH]} \\ &= 2(1 + \mathsf{len}(w)) && \text{[Algebra]} \\ &= 2(\mathsf{len}(wa)) && \text{[By Definition of } \mathsf{len}] \end{aligned}$$

This proves $\mathsf{P}(wa)$.

Thus, $\mathsf{P}(x)$ holds for all strings $x \in \Sigma^*$ by structural induction.

(b) Consider the following definition of a (rooted binary) **Tree**:
Basis Step: $\bullet$ is a **Tree**.
Recursive Step: If L is a **Tree** and R is a **Tree** then $\texttt{Tree}(\bullet, L, R)$ is a **Tree**.
The function leaves returns the number of leaves of a **Tree**. It is defined as follows:

$$\begin{aligned} \mathsf{leaves}(\bullet) &= 1 \\ \mathsf{leaves}(\texttt{Tree}(\bullet, L, R)) &= \mathsf{leaves}(L) + \mathsf{leaves}(R) \end{aligned}$$

Also, recall the definition of size on trees:

$$\begin{aligned} |\bullet| &= 1 \\ |\texttt{Tree}(\bullet, L, R)| &= 1 + |L| + |R| \end{aligned}$$

Prove that $\mathsf{leaves}(T) \geq |T|/2 + 1/2$ for all Trees T.

Solution:

For a rooted binary tree T, let $\mathsf{P}(T)$ be $\mathsf{leaves}(T) \geq |T|/2 + 1/2$. We prove $\mathsf{P}(T)$ for all rooted binary trees T by structural induction.

Base Case. We show that $\mathsf{P}(\bullet)$ holds. By definition of $\mathsf{leaves}(.)$, $\mathsf{leaves}(\bullet) = 1$ and $|\bullet| = 1$. So, $\mathsf{leaves}(\bullet) = 1 \geq 1/2 + 1/2 = |\bullet|/2 + 1/2$.

Induction Hypothesis: Suppose $\mathsf{P}(L)$ and $\mathsf{P}(R)$ hold for some arbitrary rooted binary trees L and R.

Induction Step: We prove that $\mathsf{P}(\texttt{Tree}(\bullet, L, R))$ holds.

$$\begin{aligned}
\mathsf{leaves}(\texttt{Tree}(\bullet, L, R)) &= \mathsf{leaves}(L) + \mathsf{leaves}(R) && \text{[By Definition of } \mathsf{leaves}\text{]} \\
&\geq (|L|/2 + 1/2) + (|R|/2 + 1/2) && \text{[By IH]} \\
&= (|L| + |R| + 1)/2 + 1/2 \\
&= |\texttt{Tree}(\bullet, L, R)|/2 + 1/2 && \text{[By Definition of } \mathsf{size}\text{]}
\end{aligned}$$

This proves $\mathsf{P}(\texttt{Tree}(\bullet, L, R))$.

Thus, the $\mathsf{P}(T)$ holds for all rooted binary trees T.

2. Regular Expressions

(a) Write a regular expression that matches base 10 non-negative numbers (e.g., there should be no leading zeroes).

Solution:

$$0 \cup ((1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)(0 \cup 1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)^*)$$

(b) Write a regular expression that matches all non-negative base-3 numbers that are divisible by 3.

Solution:

$$0 \cup ((1 \cup 2)(0 \cup 1 \cup 2)^* 0)$$

(c) Write a regular expression that matches all binary strings that contain the substring "111", but not the substring "000".

Solution:

$$(01 \cup 001 \cup 1^*)^*(0 \cup 00 \cup \varepsilon)111(01 \cup 001 \cup 1^*)^*(0 \cup 00 \cup \varepsilon)$$

(If you don't want the substring 000, the only way you can produce 0s is if there are only one or two 0s in a row, and they are immediately followed by a 1 or the end of the string.)