These lab problems ask you to prove some simple claims about recursively-defined string functions and concatenation. In each case, we want a self-contained proof by induction that relies on the formal recursive definitions, not on intuition. In particular, your proofs must refer to the formal recursive definition of string concatenation:

\[ w \cdot z := \begin{cases} 
    z & \text{if } w = \epsilon \\
    a \cdot (x \cdot z) & \text{if } w = ax \text{ for some symbol } a \text{ and some string } x
\end{cases} \]

You may also use any of the following facts, which we proved in class:

**Lemma 1:** Concatenating nothing does nothing: For every string \( w \), we have \( w \cdot \epsilon = w \).

**Lemma 2:** Concatenation adds length: \( |w \cdot x| = |w| + |x| \) for all strings \( w \) and \( x \).

**Lemma 3:** Concatenation is associative: \((w \cdot x) \cdot y = w \cdot (x \cdot y)\) for all strings \( w, x, \) and \( y \).

1. Let \( \#(a, w) \) denote the number of times symbol \( a \) appears in string \( w \); for example,

\[
\#(0, \text{000010101010010100}) = 12 \quad \text{and} \quad \#(1, \text{000010101010010100}) = 6.
\]

   (a) Give a formal recursive definition of \( \#(a, w) \).

   (b) Prove by induction that \( \#(a, w \cdot z) = \#(a, w) + \#(a, z) \) for any symbol \( a \) and any strings \( w \) and \( z \).

2. The reversal \( w^R \) of a string \( w \) is defined recursively as follows:

\[
w^R := \begin{cases} 
    \epsilon & \text{if } w = \epsilon \\
    x^R \cdot a & \text{if } w = a \cdot x
\end{cases}
\]

   (a) Prove that \( (w \cdot x)^R = x^R \cdot w^R \) for all strings \( w \) and \( x \).

   (b) Prove that \( (w^R)^R = w \) for every string \( w \).