1. Structural Induction
(a) Consider the following recursive definition of strings.

**Basis Step:** "" is a string

**Recursive Step:** If \( X \) is a string and \( c \) is a character then \( \text{append}(c, X) \) is a string.

Recall the following recursive definition of the function \( \text{len} \):

\[
\begin{align*}
\text{len}("") &= 0 \\
\text{len}(\text{append}(c, X)) &= 1 + \text{len}(X)
\end{align*}
\]

Now, consider the following recursive definition:

\[
\begin{align*}
\text{double}("") &= "" \\
\text{double}(\text{append}(c, X)) &= \text{append}(c, \text{append}(c, \text{double}(X)))
\end{align*}
\]

Prove that for any string \( X \), \( \text{len}(`\text{double}(X)`)) = 2\text{len}(X) \).

(b) Consider the following definition of a (binary) **Tree**:

**Basis Step:** \( \bullet \) is a **Tree**.

**Recursive Step:** If \( L \) is a **Tree** and \( R \) is a **Tree** then \( \text{Tree}(\bullet, L, R) \) is a **Tree**.

The function \( \text{leaves} \) returns the number of leaves of a **Tree**. It is defined as follows:

\[
\begin{align*}
\text{leaves}(\bullet) &= 1 \\
\text{leaves}(\text{Tree}(\bullet, L, R)) &= \text{leaves}(L) + \text{leaves}(R)
\end{align*}
\]

Also, recall the definition of \( \text{size} \) on trees:

\[
\begin{align*}
\text{size}(\bullet) &= 1 \\
\text{size}(\text{Tree}(\bullet, L, R)) &= 1 + \text{size}(L) + \text{size}(R)
\end{align*}
\]

Prove that \( \text{leaves}(T) \geq \frac{\text{size}(T)}{2} \) for all Trees \( T \).

2. Regular Expressions
(a) Write a regular expression that matches base 10 numbers (e.g., there should be no leading zeroes).

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

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