Solutions to Homework 1

Problem 1.
We prove the claim by induction on the height $n$.

- **Basis Case.** For $n = 0$ the tree consists only from one node. Therefore, the number of nodes in the tree is
  \[ 2^{n+1} - 1 = 2^0 - 1 = 2 - 1 = 1, \]
  as needed.

- **Inductive Hypothesis.** Let’s assume that for any binary tree with height up to $n$, where $n \geq 0$, the number of nodes in the tree is at most $2^{n+1} - 1$.

- **Inductive Step.** We will prove that the claim holds for any binary tree of height $n + 1$. Namely, we will prove that the number of nodes in the tree is at most $2^{(n+1)+1} - 1$.

Any binary tree of height $n + 1$ can be decomposed into two subtrees each of height $n$ and a root node as shown in Figure 1. Since each subtree has height $n$, we can apply the inductive hypothesis to each subtree and the number of nodes in a subtree is at most $2^{n+1} - 1$. Summing the nodes from the two subtrees and the root node we have that the total number of nodes is at most
  \[ 2 \cdot (2^{n+1} - 1) + 1 = 2^{n+2} - 2 + 1 = 2^{(n+1)+1} - 1, \]
  as needed.

Problem 2.
We prove the claim by induction on $n$.

- **Basis Case.** For $n = 4$ we have $n! = 1 \cdot 2 \cdot 3 \cdot 4 = 24$, and $2^4 = 16$, and thus $2^4 < n!$, as needed.

- **Inductive Hypothesis.** Let’s assume that the claim holds for any integer up to $n$, where $n \geq 4$. Namely it holds that $2^n < n!$.

- **Inductive Step.** We will prove that the claim holds for integer $n + 1$ (where $n \geq 4$). Namely, we will prove that $2^{n+1} < (n + 1)!$.

We can write $2^{n+1} = 2 \cdot 2^n$. By the inductive hypothesis we have that $2^n < n!$. Subsequently, $2 \cdot 2^n < 2 \cdot n!$. Since $2 < n$ we obtain $2 \cdot 2^n < n \cdot n!$. By the definition of the factorial operation we have that $(n + 1)! = (n + 1) \cdot n! > n \cdot n!$. Therefore, $2 \cdot 2^n < (n + 1)!$, which implies that $2^{n+1} < (n + 1)!$, as needed.
Problem 3.

We are given that for any symbol $a$, it holds
\[a^R = a,\] (1)
and for any string $u$ and symbol $a$, it holds
\[(ua)^R = au^R.\] (2)

We want to prove that for any strings $u$ and $v$ it holds that
\[(uv)^R = v^Ru^R.\]

We will prove the claim by induction on $|v|$, the length of the string $v$.

- **Basis Case.** We have $|v| = 1$, and thus $v$ is only one symbol e.g. $v = a$. Therefore, $(uv)^R = (ua)^R$. By Equation 2, we have $(ua)^R = au^R$, and by Equation 1, we have $au^R = a^Ru^R$. Since $a^R = v^R$, we obtain
  \[(uv)^R = (ua)^R = a^Ru^R = v^Ru^R,\]
as needed.

- **Inductive Hypothesis.** Let’s assume that the claim holds for any string $v$ of length at most $n$, (in other words, $|v| \leq n$). Namely, for any such string it holds
  \[(uv)^R = v^Ru^R.\]

- **Inductive Step.** We will prove the claim for any string $v$ of length equal to $n + 1$ (in other words, $|v| = n + 1$). Namely, we will prove that
  \[(uv)^R = v^Ru^R.\]

Since $|v| = n + 1$, we can write $v$ as the concatenation of one string, e.g. $w$, and a symbol, e.g. $a$, so that $v = wa$, where the string $w$ has length $|w| = n$. We have now, that
\[
(uv)^R &= (uwa)^R \\
&= a^R(uw^R) \\
&= a(uw)^R. \quad \text{(By Equation 2 applied on } uw \text{ and } a) \\
&= a(uw)^R. \quad \text{(By Equation 1)}
\]

Since the length of $w$ is $n$, we can apply the inductive hypothesis for the string $w$ and we obtain
\[(uw)^R = w^Ru^R.\]
Subsequently,

\[(uw)^R = a(uw)^R\]
\[= aw^R u^R\]
(By the inductive hypothesis on \(w\))
\[= (wa)^R u^R\]
(By Equation 2 applied on \(w\) and \(a\))
\[= (v)^R u^R\]
\[= v^R u^R,\]

as needed.

**Problem 4.**

We prove the two parts of the problem.

- First we show that if \(w \in L_1(L_2 \cap L_3)\) then \(w \in L_1 L_2 \cap L_1 L_3\).

  Since \(w \in L_1(L_2 \cap L_3)\), there must be two strings \(u\) and \(v\), such that \(u \in L_1\) and \(v \in L_2 \cap L_3\), so that \(w\) can be written as the concatenation of \(u\) and \(v\), namely \(w = uv\).

  Since \(v \in L_2 \cap L_3\), it must be that \(v \in L_2\) and \(v \in L_3\). Furthermore, since \(u \in L_1\), we get that \(uv \in L_1 L_2\) and \(uv \in L_1 L_3\). Subsequently, \(uv \in L_1 L_2 \cap L_1 L_3\), and thus, \(w \in L_1 L_2 \cap L_1 L_3\), as needed.

- We want to find languages \(L_1, L_2, L_3\), for which there is a \(w\) such that if \(w \in L_1 L_2 \cap L_1 L_3\) then \(w \notin L_1 (L_2 \cap L_3)\).

  Take

  \[
  L_1 = \{a,ab\} \\
  L_2 = \{\lambda\} \\
  L_3 = \{b\}.
  \]

  We get,

  \[
  L_1 L_2 = \{a,ab\}\{\lambda\} = \{a,ab\} \\
  L_1 L_3 = \{a,ab\}\{b\} = \{ab,abbb\},
  \]

  Subsequently,

  \[
  L_1 L_2 \cap L_1 L_3 = \{a,ab\} \cap \{ab,abbb\} = \{ab\}.
  \]

  Moreover,

  \[
  L_2 \cap L_3 = \{\lambda\} \cap \{b\} = \emptyset.
  \]
and thus
\[ L_1(L_2 \cap L_3) = \{a, ab\} \emptyset = \emptyset. \]

Take now
\[ w = ab. \]

We have
\[ ab \in L_1L_2 \cap L_1L_3 = \{ab\} \]
and
\[ ab \notin L_1(L_2 \cap L_3) = \emptyset, \]
as needed.

**Problem 5.**
See Figures 2, 3, 4, and 5.
Figure 1: Decomposition of a binary tree to two subtrees.
Figure 2: Part (a)
Figure 3: Part (b)

\[
\alpha, \alpha
\]
Figure 4: Part (c)
Figure 5: Part (d)