Implement and prove correct a function to combine two 2-3-trees of equal height, such that the inorder traversal of the resulting tree is the concatenation of the inorder traversal of the arguments, and the height of the result is either the height of the arguments, or has increased by one. Use $\textbf{if}(\text{typ } \textbf{a up},\text{typ } \textbf{a up})$ to return the result, similar to $\text{leaves}$.

```markdown
fun join :: "\textbf{a} \textbf{tree23 } \Rightarrow \textbf{a} \textbf{tree23 } \Rightarrow \textbf{a} \textbf{up},\text{typ } \textbf{a} \textbf{up}"
where
  "join Leaf Leaf = T; Leaf"

lemma join_inorder:
  fixes tl tl2 :: "\textbf{a} \textbf{tree23}"
  assumes "\textbf{height } tl \textbf{ = height } tl2"
  shows "\textbf{inorder } (\textbf{tree } (\textbf{join } tl tl2)) = (\textbf{inorder } tl @ \textbf{inorder } tl2)"
```

```
fun join :: "\textbf{a} \textbf{tree23 } \Rightarrow \textbf{a} \textbf{tree23 } \Rightarrow \textbf{a} \textbf{up},\text{typ } \textbf{a} \textbf{up}"
where
  "join Leaf Leaf = T; Leaf"

lemma join_inorder:
  fixes tl tl2 :: "\textbf{a} \textbf{tree23}"
  assumes "\textbf{height } tl \textbf{ = height } tl2"
  shows "\textbf{inorder } (\textbf{tree } (\textbf{join } tl tl2)) = (\textbf{inorder } tl @ \textbf{inorder } tl2)"
```
my (simp add delete_def inorder_def)

subsection "Balancedness"

text First a standard proof that \texttt{(const ins)} preserves \texttt{(const bal)}, \footnote{The proof is essentially the same as the proof for \texttt{bal}.}

instantiation up, \texttt{(\textit{type}height)} begin

fun height_up :: "\texttt{\textit{up}}, \texttt{nat}" where

height \texttt{(\textit{up}, \texttt{x}, \texttt{y})} = height \texttt{\textit{up}}

instance ..
end

subsection "Proofs for insert"

lemma join_bal:
  assumes \texttt{"height \textit{t1} = height \textit{t2}"}
  shows \texttt{"bal \textit{t1} \& bal \textit{t2}"}
  oops

lemma join_equal_bal:
  assumes \texttt{"height \textit{t1} = height \textit{t2}"}
  shows \texttt{"bal \textit{t1} \& bal \textit{t2}"}
  oops

lemma join
  assumes \texttt{"height \textit{t1} = height \textit{t2}"}
  shows \texttt{"bal \textit{t1} \& bal \textit{t2}"}
  oops

next
  hints
    \begin{itemize}
    \item Try to use automatic case splitting \texttt{(auto split \ldots)} instead of explicit case splitting via \texttt{case}.
    \item There will be dozens of cases.
    \item Try to find bugs in your \texttt{join} function, or isolate the case where your automatic proof does not work.
    \end{itemize}

end

text \textbf{Exercise (Joining 2-3-Trees)}

Implement and prove correct a function to combine two 2-3-trees of equal height, such that the inorder traversal of the resulting tree is the concatenation of the inorder traversal of the arguments, and the height of the result is either the height of the arguments, or has increased by one. Use \texttt{\textit{up}} to return the result, similar to \texttt{(\textit{const [source] Tree23_bal.ins})}:

fun join :: "\texttt{\textit{tree23} \& \textit{up}, \textit{tree23} \& \textit{up}, \texttt{nat}}" where

"Join Leaf Leaf = T Leaf"

lemma join_bal:
  assumes \texttt{"height \textit{t1} = height \textit{t2}"}
  shows \texttt{"bal \textit{t1} \& bal \textit{t2}"}
  oops

lemma join_equal_bal:
  assumes \texttt{"height \textit{t1} = height \textit{t2}"}
  shows \texttt{"bal \textit{t1} \& bal \textit{t2}"}
  oops

lemma join:
  assumes \texttt{"height \textit{t1} = height \textit{t2}"}
  shows \texttt{"bal \textit{t1} \& bal \textit{t2}"}
  oops
Exercise (Joining 2-3-Trees)

Implement and prove correct a function to combine two 2-3-trees of equal height, such that the inorder traversal of the resulting tree is the concatenation of the inorder traversal of the arguments, and the height of the result is either the height of the arguments, or has increased by one. Use \( \text{join} \) to return the result, similar to

\[
\text{join} \text{[const [source] Tree23_Set.imns]};
\]

```javascript
function join(t1, t2) {
  if (t1 === undefined) {
    return t2;
  } else if (t2 === undefined) {
    return t1;
  }
  // Implement join logic here
}
```

lemma join_inorder:
fixes t1 t2 :: 'a tree23'
assumes "height t1 = height t2"
assumes "bal t1 = bal t2"
shows "inorder (tree, (join t1 t2)) = (inorder t1 @ inorder t2)"
quickcheck

lemma join_bal:

proof (prove)
goal (1 subgoal):
1. inorder (tree, (join t1 t2)) = inorder t1 @ inorder t2
proof (prove):

goal (1 subgoal):
1. inorder (tree, (join t1 t2)) ⊑ inorder t1 ⊓ inorder t2

proof (prove):

goal (0 subgoal):
1. \( \forall t1, t2, t3 : b, t4. \)
   inorder (tree, (join t1 t2)) ⊑ inorder t2 ⊓ inorder t3

proof (prove):

goal (1 subgoal):
1. inorder (tree, (join t1 t2)) = inorder t1 ⊓ inorder t2
proof (prove)
goal (1 subgoal):
  1. inorder (tree, (join t1 t2)) = inorder t1 @ inorder t2

lemma join_bal:
fixes t1 t2 :: nat tree3
assumes "height t1 = height t2"
assumes "bal t1" "bal t2"
shows "inorder (tree, (join t1 t2)) = inorder t1 @ inorder t2"
oops

Text - Hints:
• Try to use automatic case splitting (-auto split: ...) instead of explicit case splitting via iar
  (There will be dozens of cases).
• To find how in more iota formulas, the case there uses automatic case does not work right.

proof (prove)
goal (1 subgoal):
  1. bal (tree, (join t1 t2)) ∧ height (join t1 t2) = height t2
lemma join_bal:
  fixes t1 t2 :: "a tree23"
  assumes \text{"height t1 = height t2"}
  shows \text{"bal (tree, (join t1 t2)) \land height (join t1 t2) = height t2"}
  using assms
  apply induction t1 t2 rules: join.induct
  apply (auto split; up; splits)
  done

proof (prove)
  goal [1 subgoal]:
  1. \text{bal (tree, (join t1 t2)) \land height (join t1 t2) = height t2}
lemma join_bal:
  fixes t1 t2 :: "a tree23"
  assumes "height t1 = height t2"
  shows "ball (tree (join t1 t2)) ∧ height (join t1 t2) = height t2"
  using assms
  apply (induction t1 t2 rule: join.induct)
  apply (auto splits: up, splits)
  done

proof (prove)
  goal (9 subgoals):
  1. height () = height () :: bal () :: bal () → bal (tree (), (), (), () ∧ height () = height ()
  2. height (t1 :: t2 :: t3 :: t4 :: t5 :: t6 :: []) = height (t1 :: t2 :: t3 :: t4 :: t5 :: t6 :: [])
    ∧ height (join t1 t2 :: t3 :: t4 :: t5 :: t6 :: []) = height t2
    using assms
    apply (induction t1 t2 rule: join.induct)
    apply (auto splits: up, splits) (* Take roughly 30 sec *)
    done

  text - Hints:
  1. Try to use automatic case splitting (auto split ...) instead of explicit case splitting via less.
proof (prove)
goal:
No subgoals!

Exercise (Bounding the Size of 2-3-Trees)

Show that for 2-3-trees, we have:

\[ \log (s(t) + 1) \leq h(t) \leq \log (|s(t)| + 1) \]

Hint: It helps to first raise the two sides of the inequality to the 2nd/3rd power.

Use sludgehammer and find-theorems to search for the appropriate lemmas.

Lemma height_bound_upper: \( h(t) \leq \log 2 \cdot (s(t) + 1) \)
Exercise: Bounding the size of 2-3 trees

Show that for 2-3 trees, we have:

\[ h(t) \leq \log_2 (|s(t)| + 1) \]

Hint: It helps to first raise the two sides of the inequality to the 2nd/3rd power.

Use sledgehammer and find-theorems to search for the appropriate lemmas.

Lemma height_bound_upper: \( \text{bal } t \rightarrow \text{height } t \leq \log_2 (|s(t)| + 1) \)

proof (prove)

goal 1 subgoal:
1. \( \text{bal } t \rightarrow \text{real } (\text{height } t) \leq \log_2 (\text{real } (|s(t)| + 1)) \)

Lemma height_bound_lower: \( \text{bal } t \rightarrow \log_2 (|s(t)| + 1) \leq \text{height } t \)

proof (prove)

using \( \log_2 \text{of power by blast} \)
proof (prove)
goal (1 subgoal):
1. bal t → log 3 (size t + 1) ≤ real (height t)
by (induction \( t \)) auto
then show \( \theta \)thesis

lemma \( \log_3 \) of power \( \leq \) log of power \( \leq \) where \( \beta \) is \( \beta \) simplified

lemma height_bound_lower:
assumes \( \beta \) \( t \)
shows \( \log_3 (\text{size } t + 1) \leq \text{height } t \)
proof
- free assms have \( \text{size } t + 1 \leq 3 ^ \text{height } t \)
  by (induction \( t \)) auto
then show \( \theta \)thesis
  using \( \log_3 \) of power \( \leq \) by fastest

proof (prove)
goal (1 subgoal):
1. \( \log_3 \) (real (size \( t \) + 1)) \( \leq \) real (height \( t \))

lemma height_bound_lower:
assumes \( \beta \) \( t \)
shows \( \log_3 (\text{size } t + 1) \leq \text{height } t \)
proof
- free assms have \( \text{size } t + 1 \leq 3 ^ \text{height } t \)
  by (induction \( t \)) auto
then show \( \theta \)thesis
  applying (fastforce) \[4 \]
  using \( \log_3 \) of power \( \leq \) by fastest

proof (prove)
using this:
size \( t + 1 \leq 3 ^ \text{height } t \)

lemma height_bound_lower:
assumes \( \beta \) \( t \)
shows \( \log_3 (\text{size } t + 1) \leq \text{height } t \)
proof
- free assms have \( \text{size } t + 1 \leq 3 ^ \text{height } t \)
  by (induction \( t \)) auto
then show \( \theta \)thesis
  applying (fastforce) \[4 \]
  using \( \log_3 \) of power \( \leq \) by fastest

proof (prove)
using this:
size \( t + 1 \leq 3 ^ \text{height } t \)

lemma height_bound_lower:
assumes \( \beta \) \( t \)
shows \( \log_3 (\text{size } t + 1) \leq \text{height } t \)
proof
- free assms have \( \text{size } t + 1 \leq 3 ^ \text{height } t \)
  by (induction \( t \)) auto
then show \( \theta \)thesis
  applying (fastforce) \[4 \]
  using \( \log_3 \) of power \( \leq \) by fastest

proof (prove)
using this:
size \( t + 1 \leq 3 ^ \text{height } t \)

lemma height_bound_lower:
assumes \( \beta \) \( t \)
shows \( \log_3 (\text{size } t + 1) \leq \text{height } t \)
proof
- free assms have \( \text{size } t + 1 \leq 3 ^ \text{height } t \)
  by (induction \( t \)) auto
then show \( \theta \)thesis
  applying (fastforce) \[4 \]
  using \( \log_3 \) of power \( \leq \) by fastest

proof (prove)
using this:
size \( t + 1 \leq 3 ^ \text{height } t \)
Lemma height_bound_lower:

assumes "bal t"

shows "\( \log 3 \ (size t + 1) \leq \text{height} t \)"

proof:

from assms have "size t + 1 \leq 3 \times \text{height} t"

apply (auto (linarith))

by (induction t) auto

then show thesis

using log3_power_le by fastforce

qed

proof (prove):

goal (1 subgoal):

1. \( \text{bal} t \to \text{Suc} \ (\text{size} t) \leq 3 \times \text{height} t \)
lemma height_bound_lower:
  assumes "bal t"  
  shows "log 3 (size t + 1) \leq height t"

proof -
  from assms have "size t + 1 \leq \ell height t"
    apply (auto [linarith!])
    by (induction t) auto
  then show ?thesis
    using log3_of_power_le by fastforce
  qed

end

proof (prove)
using this:
  bal t