Goals You learn to formalize simple situations and recursive functions in Isabelle and prove simple theorems about these.

Hint On this sheet, all proofs can be found by auto, if you use induction where necessary and give the right equations to the simplifier.


Write a function

```isabelle
sumup :: "int list ⇒ int"
```

which computes the sum of numbers of a given Isabelle/HOL list.

Evaluate the function on concrete values to get a feeling for its correctness:

```isabelle
value "sumup []"
value "sumup [1]"
value "sumup [1,2]"
value "sumup [1,2,45,62]"
```

Prove now that the \texttt{sumup} function is distributive, i.e. that \( \text{sumup} (xs \@ ys) = \text{sumup} xs + \text{sumup} ys \) holds.

Afterwards, prove that \texttt{sumup} returns the same result for reversed lists, i.e. that \( \text{sumup} (\text{rev} \: xs) = \text{sumup} \: xs \) holds

Exercise 2 [2] Nodes of binary trees

In the exercise theory, we defined a type of binary trees with functions \texttt{nleaves} and \texttt{ninner} counting the leaves and inner nodes.

Define a predicate

```isabelle
fun bt-full :: "bt ⇒ bool"
```

recognizing full binary trees. A binary tree is called \texttt{full}, iff for all \texttt{Node}s either both or none of the children are \texttt{Tip}.

Formulate and prove the the property that for each full tree holds \( \text{leaves} \: t = \text{ninner} \: t + 1 \).

\textit{Hint}: To prove this property, you will need the following lemma:

```isabelle
\( t \neq \text{Tip} \iff (\exists \: l \: v \: r . \: t = \text{Node} \: l \: v \: r) \)
```

This can be proved by case analysis on \( t \), which can be performed with \texttt{apply (cases t)}.

Exercise 3 [4] Summing Binary Trees

Write a function

```isabelle
sumtree :: "bt ⇒ int"
```

which computes the sum of all elements of a tree and a function

```isabelle
list-of-tree :: "bt ⇒ int list"
```

which computes the list of all elements (including duplicates) of a tree.

Then prove \( \text{sumtree} \: xs = \text{sumup} \: (\text{list-of-tree} \: xs) \).