Exercise 5.1 Recall the formalization of multisets from exercise 4.1. A multiset $s$ is finite iff the set of its members is finite. This is formalized by the PVS theory `finite_multiset_defs` below. That theory also defines a cardinality function `size` for finite multisets by summing up the multiplicities of the members.

1. Typecheck the theory `finite_multiset_defs`, i.e., prove the TCCs.

2. Prove that every submultiset of a finite multiset is finite.

3. Prove that the operators `empty` and `singleton` yield finite multisets.

4. Prove that the operators `union`, `add` and `remove` take finite multisets to finite multisets.

5. Prove that `intersection` yields a finite multiset if one of its arguments is finite.
   Prove that `difference` yields a finite multiset if its first argument is finite.

```plaintext
finite_multiset_defs[T : TYPE]: THEORY
BEGIN
  IMPORTING multiset_defs[T]
  IMPORTING finite_sets@finite_sets_sum_real[T]

  n: VAR nat
  x: VAR T
  s: VAR multiset[T]

  members(s): [T -> bool] = LAMBDA x: member(x,s)

  is_finite(s): bool = finite_sets_def.is_finite(members(s))

  finite_multiset: TYPE = (is_finite) CONTAINING empty[T]

  fs: VAR finite_multiset

  size(fs): nat = sum(members(fs),fs)
END finite_multiset_defs

Hint: Have a look into the prelude where finite sets are defined.
```
Exercise 5.2 In order to work with finite multisets one needs induction principles for them. Prove the following two induction principles for finite multisets:

\begin{align*}
\text{finite_multiset_induction: THEOREM} \\
\forall (P: \{\text{finite_multiset}[T] \to \text{bool}\}): \\
P(\text{empty}) \land \\
(\forall x, fs: P(fs) \implies P(\text{add}(x,fs))) \implies \\
\forall ft: P(ft) \\
\end{align*}

\begin{align*}
\text{finite_multiset_induction_union: THEOREM} \\
\forall (P: \{\text{finite_multiset}[T] \to \text{bool}\}): \\
P(\text{empty}) \land \\
(\forall x: P(\text{singleton}(x))) \land \\
(\forall fs, ft: P(fs) \land P(ft) \implies P(\text{union}(fs,ft))) \implies \\
\forall ft: P(ft) \\
\end{align*}

Hints:

- Do measure-induct and use size as the measure.
- The following lemmata\(^1\) may be helpful:

\begin{align*}
x: \text{VAR } T \\
fs, ft, ft1, ft2: \text{VAR } \text{finite_multiset}[T] \\
\end{align*}

\begin{align*}
\text{empty_or_add: LEMMA} \\
fs = \text{empty} \lor \\
(\exists x, ft: fs = \text{add}(x,ft)) \\
\end{align*}

\begin{align*}
\text{empty_or_singleton_or_union: LEMMA} \\
fs = \text{empty} \lor \\
(\exists x: fs = \text{singleton}(x)) \lor \\
(\exists ft1, ft2: ft1 \neq \text{empty} \land ft2 \neq \text{empty} \land fs = \text{union}(ft1,ft2)) \\
\end{align*}

\begin{align*}
\text{size_empty: LEMMA } & \text{size(empty) = 0} \\
\text{size_singleton: LEMMA } & \text{size(singleton(x)) = 1} \\
\text{size_union: LEMMA } & \text{size(union(fs,ft)) = size(fs) + size(ft)} \\
\text{size_add: LEMMA } & \text{size(add(x,fs)) = size(fs) + 1} \\
\end{align*}

Exercise 5.3 In exercise 4.2, you have proven that \texttt{mergesort} takes a list \texttt{xs} to a sorted list \texttt{ys}. To complete the verification of \texttt{mergesort} it remains to be shown that the list \texttt{ys} contains exactly the same members than \texttt{xs}.

1. Prove \texttt{mergesort_member: THEOREM} \texttt{member(x, mergesort(xs)) = member(x,xs)}.

\(^1\)Some of these lemmata are very hard to prove. If you do not succeed in proving them you may just use them as if they were axioms.
2. Prove \texttt{mergesort_length}: \textsc{Theorem} \( \text{length}(\text{mergesort}(\text{xs})) = \text{length}(\text{xs}) \).

**Hints:**

- Use \texttt{measure\_induct}.
- You may have to prove lemmata about how the ‘subroutines’ \texttt{split} and \texttt{merge} behave with respect to \texttt{member} and \texttt{length}.

**Exercise 5.4** Prove the following formulas using Coq.

1. \( A \lor (A \rightarrow B) \).
2. \( (A \rightarrow B) \leftrightarrow (\neg A \lor B) \).
3. \( (\forall x:X \ P(x) \lor \forall x:X \ Q(x)) \rightarrow (\forall x:X \ P(x) \lor Q(x)) \).

**Hint:** Some of these formulas are only classically valid. Check the Coq tutorial for examples how to prove classical formulas by using the law of excluded middle as an extra axiom.

**Exercise 5.5** Using the induction rule for \( \leq \) from the slides, prove the following facts in natural deduction:

- \( \forall x, y, z : \text{Nat} \ x \leq y \land y \leq z \rightarrow x \leq z \).
  \textbf{Hint:} Prove \( \forall x, y : \text{Nat} \ x \leq y \rightarrow \forall z : \text{Nat} \ (y \leq z \rightarrow x \leq z) \).

- \( \forall x, y : \text{Nat} \ x \leq y \lor y \leq x \).
  \textbf{Hint:} This is proven by \texttt{Nat\_induction} (not \( \leq \)-induction) over \texttt{Nat}.
  You need the additional lemma \( \forall x, y : \text{Nat} \ x \leq y \rightarrow \text{succ}(x) \leq y \lor x \approx y \).