Isabelle/HOL Exercises
Lists

Sets as Lists

Finite sets can obviously be implemented by lists. In the following, you will be asked to implement the set operations union, intersection and difference and to show that these implementations are correct. Thus, for a function

```
  list_union :: "['a list, 'a list] ⇒ 'a list"
```

to be defined by you it has to be shown that

```
lemma "set (list_union xs ys) = set xs ∪ set ys"
```

In addition, the functions should be space efficient in the sense that one obtains lists without duplicates (`distinct`) whenever the parameters of the functions are duplicate-free. Thus, for example,

```
lemma [rule_format]:
  "distinct xs ⟶ distinct ys ⟶ (distinct (list_union xs ys))"
```

Hint: `distinct` is defined in `List.thy`. Also the function `mem` and the lemma `set_mem_eq` may be useful.

Quantification over Sets

Define a (non-trivial) set `S` such that the following proposition holds:

```
lemma "((∀ x ∈ A. P x) ∧ (∀ x ∈ B. P x)) ⟶ (∀ x ∈ S. P x)"
```

Define a (non-trivial) predicate `P` such that

```
lemma "∀ x ∈ A. P (f x) ⟹  ∀ y ∈ f ` A. Q y"
```