# Isabelle/HOL Exercises
# Lists

## Quantifying Lists

Define a universal and an existential quantifier on lists using primitive recursion. Expression `alls P xs` should be true iff `P x` holds for every element `x` of `xs`, and `exs P xs` should be true iff `P x` holds for some element `x` of `xs`.

```
consts
  alls :: "('a ⇒ bool) ⇒ 'a list ⇒ bool"
  exs  :: "('a ⇒ bool) ⇒ 'a list ⇒ bool"
```

Prove or disprove (by counterexample) the following theorems. You may have to prove some lemmas first.

Use the `[simp]`-attribute only if the equation is truly a simplification and is necessary for some later proof.

```
lemma "alls (λx. P x ∧ Q x) xs = (alls P xs ∧ alls Q xs)"
lemma "alls P (rev xs) = alls P xs"
lemma "exs (λx. P x ∧ Q x) xs = (exs P xs ∧ exs Q xs)"
lemma "exs P (map f xs) = exs (P o f) xs"
lemma "exs P (rev xs) = exs P xs"
```

Find a (non-trivial) term `Z` such that the following equation holds:

```
lemma "exs (λx. P x ∨ Q x) xs = Z"
```

Express the existential via the universal quantifier – `exs` should not occur on the right-hand side:

```
lemma "exs P xs = Z"
```

Define a primitive-recursive function `is_in x xs` that checks if `x` occurs in `xs`. Now express `is_in` via `exs`:

```
lemma "is_in a xs = Z"
```

Define a primitive-recursive function `nodups xs` that is true iff `xs` does not contain duplicates, and a function `deldups xs` that removes all duplicates. Note that `deldups [x, y, x]` (where `x` and `y` are distinct) can be either `[x, y]` or `[y, x]`.

Prove or disprove (by counterexample) the following theorems.

**lemma** *"length (deldups xs) <= length xs"*
**lemma** *"nodups (deldups xs)"*
**lemma** *"deldups (rev xs) = rev (deldups xs)"*