Isabelle/HOL Exercises
Lists

```
primrec occurs :: "'a ⇒ 'a list ⇒ nat" where
  "occurs a [] = 0"
| "occurs a (x#xs) = (if (x=a) then Suc (occurs a xs) else occurs a xs)"

lemma [simp]:"occurs a (xs @ ys) = occurs a xs + occurs a ys "
  apply (induct xs)
  apply auto
done

lemma "occurs a xs = occurs a (rev xs)"
  apply (induct xs)
  apply auto
done

lemma "occurs a xs <= length xs"
  apply (induct xs)
  apply auto
done

lemma "occurs a (map f xs) = occurs (f a) xs"
  nitpick
⋮

lemma "occurs a (filter P xs) = (if P a then occurs a xs else 0)"
  apply (induct xs)
  apply auto
done

primrec remDups :: "'a list ⇒ 'a list" where
  "remDups [] = []"
| "remDups (x#xs) = (if (0 < occurs x xs) then (remDups xs)
                          else (x#(remDups xs)))"

lemma occurs_remDups: "occurs x (remDups xs) = min 1 (occurs x xs)"
  apply (induct xs)
  apply (auto)
done
```

```
primrec unique :: "'a list ⇒ bool" where
  "unique [] = True"
| "unique (x#xs) = (occurs x xs = 0 ∧ unique xs)"

lemma "unique(remDups xs)"
  apply (induct xs)
  apply (auto simp: occurs_remDups)
done
```