Isabelle/HOL Exercises
Lists

Counting Occurrences

Define a function *occurs*, such that *occurs x xs* is the number of occurrences of the element *x* in the list *xs*.

occurs :: "'a ⇒ 'a list ⇒ nat"

Prove (or let Isabelle disprove) the lemmas that follow. You may have to prove additional lemmas first. Use the *[simp]*-attribute only if the equation is truly a simplification and is necessary for some later proof.

lemma *"occurs a xs = occurs a (rev xs)"*
lemma *"occurs a xs <= length xs"*

Function *map* applies a function to all elements of a list: *map f* $[x_1,\ldots,x_n]$ = $[f\ x_1,\ldots,f\ x_n]$.

lemma *"occurs a (map f xs) = occurs (f a) xs"*

Function *filter :: ('a ⇒ bool) ⇒ 'a list ⇒ 'a list* is defined by

```
filter P [] = []
filter P (x # xs) = (if P x then x # filter P xs else filter P xs)
```

Find an expression *e* not containing *filter* such that the following becomes a true lemma, and prove it:

lemma *"occurs a (filter P xs) = e"*

With the help of *occurs*, define a function *remDups* that removes all duplicates from a list.

remDups :: "'a list ⇒ 'a list"

Find an expression *e* not containing *remDups* such that the following becomes a true lemma, and prove it:

lemma *"occurs x (remDups xs) = e"*

With the help of *occurs* define a function *unique*, such that *unique xs* is true iff every element in *xs* occurs only once.

unique :: "'a list ⇒ bool"

Show that the result of *remDups* is *unique*.