Counting Occurrences

Define a function \texttt{occurs}, such that \texttt{occurs} \( x \) \( xs \) is the number of occurrences of the element \( x \) in the list \( xs \).

\texttt{occurs} :: "'a ⇒ 'a list ⇒ nat"

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

\texttt{lemma} "occurs a xs = occurs a (rev xs)"
\texttt{lemma} "occurs a xs <= length xs"

Function \texttt{map} applies a function to all elements of a list: \texttt{map f [x₁,\ldots,xₙ]} = \texttt{[f x₁,\ldots,f xₙ]}.

\texttt{lemma} "occurs a (map f xs) = occurs (f a) xs"

Function \texttt{filter} :: ('a ⇒ bool) ⇒ 'a list ⇒ 'a list is defined by

\texttt{filter P [] = []}
\texttt{filter P (x # xs) = (if P x then x # filter P xs else filter P xs)}

Find an expression \( e \) not containing \texttt{filter} such that the following becomes a true lemma, and prove it:

\texttt{lemma} "occurs a (filter P xs) = e"

With the help of \texttt{occurs}, define a function \texttt{remDups} that removes all duplicates from a list.

\texttt{remDups} :: "'a list ⇒ 'a list"

Find an expression \( e \) not containing \texttt{remDups} such that the following becomes a true lemma, and prove it:

\texttt{lemma} "occurs x (remDups xs) = e"

With the help of \texttt{occurs} define a function \texttt{unique}, such that \texttt{unique xs} is true iff every element in \( xs \) occurs only once.

\texttt{unique} :: "'a list ⇒ bool"

Show that the result of \texttt{remDups} is \texttt{unique}. 