Exercise 1.4 Adding elements to the end of a list

Recall the definition of lists from the lecture. Define a function `suc` that appends an element at the right end of a list. Do not use the existing append operator `@` for lists.

```haskell
fun suc :: "a list ⇒ 'a ⇒ 'a list"
```

Convince yourself on some test cases that your definition of `suc` behaves as expected, for example run:

```haskell
value "suc [] c"
```

Also prove that your test cases are indeed correct, for instance show:

```haskell
lemma "suc [] c = [c]"
```

Next define a function `reverse` that reverses the order of elements in a list. (Do not use the existing function `rev` from the library.) Hint: Define the reverse of `@` as using the `suc` function.

```haskell
fun reverse :: "a list ⇒ 'a list"
```

```haskell
end
```
Recall the definition of lists from the lecture. Define a function \texttt{snoc} that appends an element at the right end of a list. Do not use the existing append operator @ for lists.

\texttt{fun snoc :: \texttt{\texttt{\textasciicircum} \texttt{\textasciicircum} \texttt{\textasciicircum}} \Rightarrow \texttt{\textasciicircum} \Rightarrow \texttt{\textasciicircum}}

Convince yourself on some test cases that your definition of \texttt{snoc} behaves as expected, for example run:

\texttt{value \texttt{\texttt{\textasciicircum} \texttt{\textasciicircum} \texttt{\textasciicircum}} \texttt{\texttt{\textasciicircum}}} \texttt{\texttt{\textasciicircum}}

Also prove that your test cases are indeed correct, for instance show:

\texttt{lemma \texttt{\texttt{\textasciicircum} \texttt{\textasciicircum} \texttt{\textasciicircum}} \texttt{\texttt{\textasciicircum}} \texttt{\texttt{\textasciicircum}}}

Next define a function \texttt{reverse} that reverses the order of elements in a list. (Do not use the existing function \texttt{rev} from the library.) Hint: Define the reverse of \texttt{x @ xs} using the \texttt{snoc} function.

\texttt{fun reverse :: \texttt{\textasciicircum} \Rightarrow \texttt{\textasciicircum}}

Demonstrate that your definition is correct by running some test cases, and proving that those test cases are correct. For example:

\texttt{value \texttt{\textasciicircum} \texttt{\textasciicircum} \texttt{\textasciicircum} \texttt{\textasciicircum} \texttt{\textasciicircum} \texttt{\textasciicircum}}
Convince yourself on some test cases that your definition of \texttt{suc} behaves as expected.

for example run:

\begin{verbatim}
value "suc []" 
\end{verbatim}

Also prove that your test cases are indeed correct, for instance show:

\begin{verbatim}
lemma "suc [] = [s]
\end{verbatim}

Next define a function \texttt{reverse} that reverses the order of elements in a list. (Do not use the existing function \texttt{rev} from the library.) Hint: Define the reverse of \( x \# xs \) as using the \texttt{suc} function.

\begin{verbatim}
fun reverse :: "a list \Rightarrow "a list"

\end{verbatim}

Demonstrate that your definition is correct by running some test cases, and proving that those test cases are correct. For example:

\begin{verbatim}
value "reverse [a, b, c]" 

\end{verbatim}

\begin{verbatim}
lemma "reverse [a, b, c] = [c, b, a]"

\end{verbatim}

Prove the following theorem. Hint: You need to find an additional lemmas relating \texttt{reverse} and \texttt{suc} to prove it.

\texttt{theorems} "reverse (reverse xs) = xs"
proof (prove)
goal (1 subgoal):
1: reverse [snoc xs x] = x # reverse xs

proof (prove)
goal (2 subgoals):
1. reverse [snoc xs x] = x
2. As xs, reverse [reverse xs] = xs ==> reverse (reverse [a # xs]) = a # xs