Merge Sort

Sorting with lists

For simplicity, we sort natural numbers.

Define a predicate \textit{sorted} that checks if each element in the list is less or equal to the following ones; \texttt{le n xs} should be true iff \texttt{n} is less or equal to all elements of \texttt{xs}.

\begin{verbatim}
consts
  le    :: "nat ⇒ nat list ⇒ bool"
  sorted :: "nat list ⇒ bool"
\end{verbatim}

Define a function \texttt{count xs x} that counts how often \texttt{x} occurs in \texttt{xs}.

\begin{verbatim}
consts
  count :: "nat list ⇒ nat ⇒ nat"
\end{verbatim}

Merge sort

Implement \textit{merge sort}: a list is sorted by splitting it into two lists, sorting them separately, and merging the results.

With the help of \texttt{recdef} define two functions

\begin{verbatim}
consts merge :: "nat list × nat list ⇒ nat list"
  msort :: "nat list ⇒ nat list"
\end{verbatim}

and show

\begin{verbatim}
theorem "sorted (msort xs)"
theorem "count (msort xs) x = count xs x"
\end{verbatim}

You may have to prove lemmas about \texttt{ex.sorted} and \texttt{count}.

Hints:

- For \texttt{recdef} see Section 3.5 of the Isabelle/HOL tutorial.

- To split a list into two halves of almost equal length you can use the functions \texttt{n div 2}, \texttt{take} and \texttt{drop}, where \texttt{take n xs} returns the first \texttt{n} elements of \texttt{xs} and \texttt{drop n xs} the remainder.
• Here are some potentially useful lemmas:
  \texttt{linorder\_not\_le}: (\neg x \leq y) = (y < x)
  \texttt{order\_less\_le}: (x < y) = (x \leq y \land x \neq y)
  \texttt{min\_def}: \texttt{min }a\texttt{ b }= \texttt{(if }a\texttt{ \leq b then }a\texttt{ else }b\texttt{)}