Exercise 6.1 Selection Sort

Selection sort (also known as MinSort) sorts a list by repeatedly moving the smallest element of the remaining list to the front.

Define a function that takes a non-empty list, and returns the minimum element and the list with the minimum element removed

fun find_min :: "a::linorder list ⇒ a × a list"

Show that find_min returns the minimum element

lemma find_min_min:
  assumes "find_min xs = (y,ys)"
  assumes "xs≠[]"
  shows "a∈set xs ⇒ y ≤ a"

Show that find_min returns exactly the elements from the list

lemma find_min_mset:
  assumes "find_min xs = (y,ys)"
  assumes "xs≠[]"
  shows "mset xs = add_mset y (mset ys)"

Show the following lemma on the length of the returned list, and register it as \([dest]\). The function package will require this to show termination of the selection sort function

lemma find_min_snd_len_decr[dest]:
  assumes "(y,ys) = find_min (x#xs)"
  shows "length ys < length (x#xs)"

Selection sort can now be written as follows:

fun sel_sort where
  "sel_sort [] = []" |
  "sel_sort xs = (let (y,ys) = find_min xs in y#sel_sort ys)"

Show that selection sort is a sorting algorithm:

lemma sel_sort_mset[simp]: "mset (sel_sort xs) = mset xs"
lemma "sorted (sel_sort xs)"
Define cost functions for the number of comparisons of \texttt{find\_min} and \texttt{sel\_sort}.

\begin{verbatim}
fun c\_find\_min :: "'a list ⇒ nat"

fun c\_sel\_sort

Try to find a closed formula for \texttt{c\_sel\_sort}! If you do not succeed, try to find a good estimate. (Hint: Should be \(O(n^2)\))

To find a closed formula: On paper:

\begin{itemize}
\item Put up a recurrence equation (depending only on the length of the list)
\item Solve the equation (Assume that the solution is an order-2 polynomial) In Isabelle:
\item Insert the solution into the lemma below, and try to prove it
\end{itemize}

\textbf{Homework 6} Quicksort

\textit{Submission until Friday, 2. 6. 2017, 11:59am.} We extend the notion of a sorting algorithm, by providing a key function that maps the actual list elements to a linearly ordered type. The elements shall be sorted according to their keys.

\begin{verbatim}
fun sorted\_key :: "('a ⇒ 'b::linorder) ⇒ 'a list ⇒ bool" where
  "sorted\_key k [] = True"
| "sorted\_key k (x # xs) = ((∀ y∈set xs. k x ≤ k y) & sorted\_key k xs)"
\end{verbatim}

Quick sort can be defined as follows: (Note that we use \texttt{nat} for the keys, as this causes less trouble when writing Isar proofs than a generic \texttt{'b::linorder})

\begin{verbatim}
fun qsort :: "('a ⇒ nat) ⇒ 'a list ⇒ 'a list" where
  "qsort k [] = []"
| "qsort k (p#xs) = qsort k [x←xs. k x<k p]@p#qsort k [x←xs. ¬k x<k p]"
\end{verbatim}

The syntax \texttt{filter P xs} is a shortcut notation for \texttt{filter P xs}.

Show that quicksort is a sorting algorithm:

\begin{verbatim}
lemma qsort\_preserves\_mset: "mset (qsort k xs) = mset xs"
lemma qsort\_sorts: "sorted\_key k (qsort k xs)"
\end{verbatim}

The following is a cost function for the comparisons of quick sort:

\begin{verbatim}
fun c\_qsort :: "('a ⇒ nat) ⇒ 'a list ⇒ nat" where
  "c\_qsort k [] = 0"
| "c\_qsort k (p#xs)
  = c\_qsort k [x←xs. k x<k p] + c\_qsort k [x←xs. k x≥k p] + 2*length xs"
\end{verbatim}

Show that the number of required comparisons is at most \((\text{length } xs)^2\).

Hints:
• Do an induction on the length of the list, and, afterwards, a case distinction on the list constructors.
• It might be useful to prove $a^2 + b^2 \leq (a+b)^2$ for $a, b :: \text{nat}$
• Have a look at the lemma sum_length_filter_compl

lemmas length_induct_rule = measure_induct_rule[where f=length, case_names shorter]

lemma "c_\text{qsort} \ k \ \text{xs} \ \leq (\text{length} \ \text{xs})^2"
proof (induction \text{xs} rule: length_induct_rule)
case (shorter \text{xs}) thm shorter.IH
  show ?case proof (cases \text{xs})
  case Nil
    then show ?thesis by auto
  next
  case (Cons \ x \ \text{xs}')

  Insert your proof here
qed
qed

For 3 bonus points, show that quicksort is stable. You will probably run into subgoals containing terms like $[x \leftarrow \text{xs} . \ k \ x < k \ p \ \land \ k \ x = a]$. Try to find a simpler form for them. (Cases on $a < k \ p$!)

lemma qsort_stable: "$[x \leftarrow \text{qsort} \ k \ \text{xs} . \ k \ x = a] = [x \leftarrow \text{xs} . \ k \ x = a]$"