Isabelle/HOL Exercises
Advanced

Merge Sort

Sorting with lists

For simplicity we sort natural numbers.

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

```
consts
  le     :: "nat ⇒ nat list ⇒ bool"
  sorted :: "nat list ⇒ bool"
```

Define a function *count xs x* that counts how often *x* occurs in *xs*.

```
consts
  count :: "nat list => nat => nat"
```

Merge sort

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

With the help of *recdef* define two functions

```
consts merge :: "nat list × nat list ⇒ nat list"
       msort :: "nat list ⇒ nat list"
```

and show

```
theorem "sorted (msort xs)"
theorem "count (msort xs) x = count xs x"
```

You may have to prove lemmas about *ex.sorted* and *count*.

Hints:

- For *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 *n div 2*, *take* und *drop*, where *take n xs* returns the first *n* elements of *xs* and *drop n xs* the remainder.

- Here are some potentially useful lemmas:
  linorder_not_le: $(\neg\ x \leq y) = (y < x)$
  order_less_le: $(x < y) = (x \leq y \wedge x \neq y)$
  min_def: `min a b = (if a` $\leq$ `b then a else b)`