Functional Programming
Exercises Week 8
(for December 2, 2016)

1. Exercise 6.11

2. Consider the function `concat : ('a list) list -> 'a list` defined by

```
let rec concat = function
  | []        -> []
  | xs :: xss -> xs @ concat xss
```

Prove that for any lists of lists *xss* and *yss*,

$$\texttt{concat}\ (xss\ \texttt{@}\ yss) = \texttt{concat}\ xss\ \texttt{@}\ \texttt{concat}\ yss$$

by structural induction on *xss*. What will happen if you do induction on *yss*? *Hint:* Recall the definition of `@`:

```
let rec (@) xs ys = match xs with []      -> ys
                                | x::xs -> x::(xs @ ys)
```

You may use the equation

$$(xs\ \texttt{@}\ ys)\ \texttt{@}\ zs = xs\ \texttt{@}\ (ys\ \texttt{@}\ zs) \qquad (\star)$$

3. Give the induction scheme for type `'a ltree`, which is defined as follows:

```
type 'a ltree = Leaf of 'a | Node of ('a ltree * 'a ltree)
```

4. Define a function `flatten_ltree : 'a ltree -> 'a list` that produces the list of the labels on the leafs in left-to-right order; for example,

```
flatten_ltree (Node(Node(Leaf 1, Node(Leaf 2, Leaf 3)), Leaf 1))
```

should evaluate to `[1;2;3;1]`.

5. Define a function `join_ltree : ('a ltree) ltree -> 'a ltree` that replaces each leaf of the given tree by the tree that is stored in the leaf.

6. ($\star$) Prove by structural induction that for any t of type `'a ltree`,

$$\texttt{flatten\_ltree}\ (\texttt{join\_ltree}\ t) = \texttt{concat}\ (\texttt{map}\ \texttt{flatten\_ltree}\ (\texttt{flatten\_ltree}\ t))$$