Functional Programming
Exercises Week 8
(for December 4, 2009)

1. Read Chapters 6 and 7 of the lecture notes.

2. Use induction over lists to prove the equation

$$\texttt{sumlen}\ xs = (\texttt{sum}\ xs, \texttt{length}\ xs)$$

using the function definitions

```
let rec sum = function []    -> 0
                     | x::xs -> x + sum xs

let rec length = function []    -> 0
                        | _::xs -> 1 + length xs

let rec sumlen = function
  | []    -> (0,0)
  | x::xs -> let (s,l) = sumlen xs in (s+x,l+1)
```

3. Give a tail recursive implementation of the function `length : 'a list -> int`, computing the length of a list.

4. Use induction over lists to prove that your function from Exercise 3, produces the same results as the non tail recursive one given in Exercise 2.

5. Use tupling to implement a more efficient version of the function `split_at`:

```
let rec take n xs = if n < 1 then [] else match xs with
  | []    -> []
  | x::xs -> x :: take (n-1) xs

let rec drop n xs = if n < 1 then xs else match xs with
  | []    -> []
  | _::xs -> drop (n-1) xs

let split_at n xs = (take n xs,drop n xs)
```

6. Find a non tail recursive function in the modules from the lecture that has not been treated yet. Justify why it is not tail recursive.