Assignment 3
Advanced Functional Programming
Topic: Specification-based Testing
Issued on: 04/21/2010, due date: 04/27/2010

For this assignment a Haskell script named `AssFFP3.hs` shall be written offering functions which solve the problems described below. This file `AssFFP3.hs` shall be stored in your home directory, as usual on the top most level. Comment your programs meaningfully. Use constants and auxiliary functions, where appropriate. In Haskell, an editor buffer can be realized as a string together with the position of a cursor as shown below:

```
type Buffer = (Int,String)
empty   :: Buffer                    -- the empty buffer
insert  :: Char -> Buffer -> Buffer -- insert character before cursor
delete  :: Buffer -> Buffer          -- delete character before cursor
left    :: Buffer -> Buffer          -- move cursor left one character
right   :: Buffer -> Buffer          -- move cursor right one character
atLeft  :: Buffer -> Bool            -- is cursor at left end?
atRight :: Buffer -> Bool            -- is cursor at right end?
```

- Provide implementations for the above functions of an editor buffer.
- A more efficient implementation of the editor buffer is possible if the data type

  ```
  type BufferI = (String,String)
  ```

  is used instead of `Buffer`. We here assume that the first string gives in reverse order the characters up to (but not including) the one at the cursor position, the second string the characters starting at (and including) the character at the cursor position.

  Provide implementations for the above buffer functions on the new data type `BufferI`, which shall be denoted `emptyI`, `insertI`, etc.
- Define a Haskell function `retrieve` with signature `retrieve :: BufferI -> Buffer`, which maps a buffer given in "efficient" representation to the equivalent buffer in the standard representation.
- Check, using `QuickCheck`, the correctness of the implementation of the buffer functions on `BufferI` with respect to the corresponding functions on `Buffer`. We consider the latter as the specification of the buffer operations for this purpose. Define similarly to the example on queues in lecture part 2 for each of the above functions one (or more) properties in your Haskell program allowing to check the desired equivalence of functions. Regarding names, please follow the naming convention used in the example on queues from lecture part 4.