Assignment 1
Advanced Functional Programming
Topics: Higher-order functions
Issued on: 03/19/2008, due date: 04/28/2008

For this assignment a Haskell script named `AssFFP1.hs` shall be written offering functions which solve the problems described below. This file `AssFFP1.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.

- Higher-order functions such as `map`, `filter`, and `foldl` demonstrate the usefulness of allowing functions as arguments of other functions. As an exercise we here want to extend our portfolio of higher-order functions by a few additional instances allowing us in the future to conveniently deal with various variants of iteration.

  To this end, develop Haskell functions `for`, `while`, `repeat`, and `loop` with the signatures

  - `for :: (a -> a) -> Int -> a -> a`
  - `while :: (a -> Bool) -> (a -> a) -> a -> a`
  - `repeat :: (a -> a) -> (a -> Bool) -> a -> a`
  - `loop :: (a -> a) -> (a -> Bool) -> (a -> a) -> a -> a`

  which shall be defined as follows:

  - Applied to arguments `f`, `n`, and `z`, the expression `for f n z` shall evaluate to $f^n(z)$, if `n` is positive; otherwise to `z`.
  - Applied to arguments `b`, `f`, and `z`, the expression `while b f z` evaluates to `z`, if `b z` evaluates to `False`; otherwise to `while b f (f z)`.
  - Applied to arguments `f`, `b`, and `z`, the expression `repeat f b z` evaluates to `f z`, if `(b . f) z` evaluates to `True`; otherwise to `repeat f b (f z)`.
  - Applied to arguments `f`, `b`, `g` and `z`, the expression `loop f b g z` evaluates to `f z`, if `(b . f) z` evaluates to `True`; otherwise to `loop f b g ((g . f) z)`.