Functional programming

Exercises IV

Lazy evaluation

1. Identify the redexes in the following expressions, and determine whether each redex is innermost, outermost, neither, or both:

   ```
   1 + (2*3)
   (1+2) * (2+3)
   fst (1+2, 2+3)
   (\x -> 1 + x) (2 * 3)
   ```
2. Show why outermost evaluation is preferable to innermost evaluation for purposes of evaluating the expression fst (1+2, 2+3).
3. Given the definition mult = \x -> (\y -> x*y) show how the evaluation of mult 2 3 can be broken down into four separate steps
4. Using a list comprehension, define an expression fibs :: [Integer] that generates the infinite sequence of Fibonacci numbers
   0, 1, 1, 2, 3, 5, 8, 13, 21, 34, …
   using the following simple procedure:
   a. The first two numbers are 0 and 1
   b. The next is the sum of the previous two
   c. Return to the second step

   Hint: make use of the library functions zip and tail. Note that numbers in the Fibonacci sequence quickly become large, hence the use of the type Integer of arbitrary-precision integers above.
5. Define appropriate versions of the library functions:

   ```
   repeat :: a -> [a]
   repeat x = xs where xs = x : xs

   take :: Int -> [a] -> [a]
   take 0 _ = []
   take _ [] = []
   take n (x:xs) = x : take (n-1) xs

   replicate :: Int -> a -> [a]
   replicate n = take n . repeat
   ```

   for the following type of binary trees:

   ```
   data Tree a = Leaf | Node (Tree a) a (Tree a) deriving Show
   ```
6. Newton's method for computing the square root of a (non-negative) floating point number n can be expressed as follows:
   a. Start with an initial approximation to the result
   b. Given the current approximation a, the next approximation is defined by the function next a = (a + n/a) / 2

c. Repeat the second step until the two most recent approximations are within some desired distance of one another, at which point the most recent value is returned as the result

Define a function sqroot :: Double -> Double that implements this procedure. Hint: first produce an infinite list of approximations using the library function iterate. For simplicity, take the number 1.0 as initial approximation, and 0.00001 as the distance value.