Funktionale Programmierung (ALP 1), WS 2011/12 — 3. Übungsblatt

Test ab 7. November 2011

Bei allen Funktionen ist *generell* eine Typdeklaration anzugeben.

12. Gültigkeitsbereiche (10+10 Punkte)

    Geben Sie für jeden eingeführten Namen den Gültigkeitsbereich an.

    (a)
    ```
    x = 25:: Integer
    biggerThanAVG3:: Integer->Integer->Integer->Integer
    biggerThanAVG3 x y z =
       sum (map (\x -> if fromIntegral x > avg3 x y z then 1 else 0)
                [x ,y, z])
       where avg3:: Integer->Integer->Integer->Double
             avg3 a b c = fromIntegral (a+b+c) / 3
    test1:: Integer
    test1 = biggerThanAVG3 3 4 5
    fläche:: Float -> Float
    fläche x = 2*x^2*pi -- Fläche eines Kreises mit Radius x
    ```

    (b)
    ```
    f x y =
       let n = 3 in take n (g y) ++ take n (g x)
       where g x = take n xys
                where
                  xys = [x] ++ yxs
                  yxs = [y] ++ xys
             n = 10
    ```

13. (10 Punkte) Schreiben Sie eine Funktion, die berechnet, wie viele Zahlen in einer Liste von `Integer`-Werten *kleiner* als der Durchschnittswert sind.

14. Preisberechnung (10 Punkte)

    Listen der beiden folgenden Datentypen

    ```
    type Einkaufsliste = [(String, Float)]
    type Preisliste    = [(String, Float)]
    ```

    geben an, *was* gekauft werden soll und *vieviel* (in einer passenden Einheit, z. B. kg), zum Beispiel `[("Mehl",0.5), ("Butter",0.25)]`, und andererseits den Preis in Euro pro Einheit für jeden Artikel.

    (a) Definieren Sie eine Funktion

    ```
    preis:: Preisliste -> Einkaufsliste -> (Float,[String])
    ```

    zur Berechnung des Gesamtpreises aller Artikel einer Einkaufsliste. Die zweite Komponente des Ergebnisses soll die Liste der Artikelnamen enthalten, die in der Preisliste nicht gefunden wurden.

    (b) Erweitern Sie die Funktion so, dass der Preis für jeden gekauften Artikel der Einkaufsliste auf Cent gerundet wird.

15. Funktionsiteration (10 Punkte)

Die folgende Funktion `iter` wendet eine Funktion f n-mal hintereinander auf ein Argument x an. Zum Beispiel liefert `iter 3 f x` das Ergebnis $f(f(f(x)))$, das man in der Mathematik manchmal als $f^3(x)$ oder noch genauer $f^{(3)}(x)$ schreibt, damit man es nicht mit der Potenz $(f(x))^3$ verwechselt.

```
iter n f x
   | n==0  = x
   | n>0   = f (iter (n-1) f x)
```

(a) Welchen Typ hat `iter`?

(b) Geben Sie eine alternative Definition als Funktion `iter n f` ohne den Parameter x.

(c) Lösen Sie Aufgabe 4b (Zinseszinsen) mit Hilfe der Funktion `iter` und der Lösung von Aufgabe 4a. (Achten Sie auf die Reihenfolge der Argumente.)

16. Iterierter Logarithmus (10 Punkte)

Die Funktion `logBase a x` $= \log_a x$ berechnet den Logarithmus von x zur Basis a; das ist die Zahl y, für die $a^y = x$ ist.

(a) Bestimmen Sie die kleinste Zahl x, für die `logBase 2 x >= 5` ist.

(b) Bestimmen Sie die kleinste Zahl x, für die `iter 3 (logBase 2) x >= 2` ist.

17. Potenzieren, Summe und Produkt (10 Punkte)

Das Potenzieren mit einer natürlich Zahl als Exponent kann man als iteriertes Multiplizieren definieren: $x^n := x \cdot x \cdot x \cdots x$ mit n Faktoren:

```
potenz x n = iter n (x*) 1
```

(a) Die "Turmfunktion" `turm x k` $= x \uparrow k := x^{x^{\cdot^{\cdot^{\cdot^{x}}}}}$ ist eine geschachtelte Potenz, bei der in dem Turm auf der rechten Seite x insgesamt k-mal vorkommt. (Potenzieren ist rechtsassoziativ!) Definieren Sie diese Funktion unter Verwendung von `potenz` und `iter`.

(b) Die *Multiplikation* kann als iterierte Addition aufgefasst werden. Schreiben Sie eine entsprechende Funktion `mal a b`, die analog zur Funktion `potenz` das Produkt als iterierte Summe definiert.

(c) Welche Funktion muss man iterieren, damit man die *Summenfunktion* `plus a b` $= a + b$ erhält? Schreiben Sie eine entsprechende Funktionsdefinition für `plus`.

18. Funktionsiteration (10 Punkte)

(a) Jemand hat die Funktion `g = iter 23` definiert. Wie können Sie das Argument 23 aus der Funktion `g` herausfinden? Schreiben Sie eine Funktion `entdecke`, die für alle $n \geq 0$ die folgende Beziehung erfüllt:

`entdecke (iter n) == n`,

(b) Addition: Schreiben Sie eine Funktion `sumiter` mit der Eigenschaft

`sumiter (iter a) (iter b) == iter (a+b)`, für alle $a, b \geq 0$.

Verwenden Sie nicht einfach `entdecke` und `+`, sondern suchen Sie eine direktere Definition.

(c) Multiplikation: Schreiben Sie eine analoge Funktion `proditer` für das Produkt von a und b.