WS 2017/2018

Funktionale Programmierung

6. Übungsblatt

Prof. Dr. Margarita Esponda

(Abgabetermin: Do. den 7.12., um 11:50, Raum 161)

Ziel: Auseinandersetzung mit algebraischen Datentypen und Funktionen höherer Ordnung.

1. Aufgabe (4 Punkte)

In einer Liste aus vergleichbaren Elementen wird ein lokales Maximum als das Element definiert, das streng grösser ist als die Elemente, die unmittelbar davor und danach stehen. Definieren Sie eine polymorphe Funktion, die bei Eingabe einer Liste die Liste aller lokalen Maxima berechnet.

Anwendungsbeispiele:

listOfLocalMaxs [2, 4, 5, 1, 6, 5, 4, 3, 2, 7] => [5, 6]
listOfLocalMaxs [1.0, 2.0, 3.0, 3.0, 4.0, 5.0] => []
listOfLocalMaxs "local maximum" => "olmxu"

2. Aufgabe (8 Punkte)

Definieren Sie eine polymorphe Funktion **largestSeq**, die die längste sich wiederholende Sequenz aus einer Liste findet (siehe Vorlesungsfolien).

Anwendungsbeispiele:

largestSeq "z**aabbc**abcabc**aabbc**zuabc" => "aabbc"

largestSeq [7,2,3,1,0,0,2,3,1,0,1,1,0,1,3] => [2,3,1,0]

Analysieren Sie die Komplexität der Funktion.

3. Aufgabe (8 Punkte)

Nehmen wir an, wir müssen eine Reihe von Berechnungen realisieren mit Zahlen, die folgende Form haben:

$a + b\sqrt{3}$, wobei ***a*** und ***b*** ganze Zahlen sind.

Würden wir zuerst die Wurzeln ausrechnen und dann die Summen machen, hätten wir Rundungsfehler, die im Laufe der Berechnungen größer werden können. Die Rundungsfehler können minimiert werden, wenn wir zuerst nur die ganzzahligen Operationen realisieren und das Ausrechnen der Wurzeln ans Ende verschieben.

Beispiel:

$$\left(3+2\cdot\sqrt{3}\right)\cdot\left(2+\sqrt{3}\right)=12+7\cdot\sqrt{3}$$

a) Definieren Sie einen algebraischen Datentyp **Root3Num**, der unsere Zahlen mit Hilfe der Koeffizienten **a** und **b** darstellt, und definieren Sie die Additions-, Subtraktions- und Multiplikationsoperation für diesen Datentyp.

b) Zum Testen definieren Sie eine Funktion **getValue,** die eine **Root3Num** Variable in einer Gleitkommazahl ausrechnet.

c) Deklarieren Sie Ihren **Root3Num** Typ als Instanz der **Eq** und **Show** Klasse, indem Sie auch die Funktionen selber definieren, die dafür notwendig sind.

4. Aufgabe (6 Punkte)

Definieren Sie eine Funktion **weekday** in Haskell, die bei Eingabe eines Datums den Wochentag mit Hilfe folgender Formeln des Gregorianischen Kalenders berechnet.

Die Formel **weekday** berechnet eine ganze Zahl zwischen 0 (Sonntag) und 6 (Samstag).

$$y_0 = year - \frac{14 - month}{12}$$

$$x = y_0 + \frac{y_0}{4} - \frac{y_0}{100} + \frac{y_0}{400}$$

$$m_0 = month + 12\left(\frac{14 - month}{12}\right) - 2$$

$$weekday = \mathrm{mod}\left(\left(day + x + \frac{(31 \cdot m_0)}{12}\right), 7\right)$$

Verwenden Sie in der Funktionsdefinition den algebraischen Datentyp **Weekday** aus der Vorlesung und definieren Sie einen eigenen algebraischen Datentyp **Month** für die Monatsnamen.

Sie müssen in Ihrer Funktion kontrollieren, dass der Wertebereich des Eingabedatums korrekt ist, und die **error**-Funktion für falsche Eingaben verwenden.

Anwendungsbeispiel:

weekday 23 Nov 2017 => Th

weekday 29 Feb 2017 => *** Exception: incorrect date! .....

5. Aufgabe (8 Punkte)

Betrachten Sie folgende algebraische Datentypen:

```
data B = T | F deriving Show
data Nat = Zero | S Nat  deriving Show
```

a) Definieren Sie folgende Funktionen damit:

```
eqB :: B -> B -> B
xorB :: B -> B -> B
addN  :: Nat -> Nat -> Nat  -- endrekursive Funktionsdefinition für die Summe
minN :: Nat -> Nat -> Nat   -- Minimum Nat
isTeiler :: Nat -> Nat -> B  -- überprüft, ob die zweite Zahl die erste Zahl teilt
ggtN :: Nat -> Nat -> Nat   -- größter gemeinsamer Teiler
```

b) Definieren Sie unter Verwendung der ***foldn*** Funktion aus der Vorlesung folgende Funktionen:

```
powN :: Nat -> Nat -> Nat   -- Potenz für natürliche Zahlen
subN :: Nat -> Nat -> Nat   -- Subtraktion (subN a b => Zero, wenn b>a)
```

Wichtige Hinweise:

3. Schreiben Sie in alle Funktionen die entsprechende Signatur.
4. Verwenden Sie geeignete Namen für Ihre Variablen und Funktionsnamen, die den semantischen Inhalt der Variablen oder die Semantik der Funktionen wiedergeben.
5. Verwenden Sie vorgegebene Funktionsnamen, falls diese angegeben werden.
6. Kommentieren Sie Ihre Programme.
7. Verwenden Sie geeignete lokale Funktionen und Hilfsfunktionen in Ihren Funktionsdefinitionen.