WiSe 2015/2016

Funktionale Programmierung

7. Übungsblatt (Abgabe: Mi., den 09. Dez. um 10:10 Uhr)

Prof. Dr. Margarita Esponda

Ziel: weitere Auseinandersetzung mit algebraische Datentypen und Typ-Klassen.

1. Aufgabe (8 Punkte)

a) Definieren Sie eine Funktion **compress**, die eine Liste von Bits bekommt und diese in kompakter Form zurückgibt, indem sie nebeneinander stehende gleiche Bits mit einer Zahl zusammenfasst. Definieren Sie zuerst einen algebraischen Datentyp **Bits** dafür.

Damit aus den komprimierten Bits wieder die originale Bitsequenz hergestellt werden kann, wird am Anfang der Ergebnisliste eine 0 eingetragen, wenn die Sequenz kein Null (Zero) am Anfang hat. D.h. es wird davon ausgegangen, dass die erste Zahl die Anzahl der Nullen entspricht.

Anwendungsbeispiele:

compress [One, Zero, Zero, One, One, One, One] => [0,1,2,4]

compress [Zero, Zero, Zero, Zero, One, One, Zero, Zero, Zero, Zero] => [4,2,4]

b) Programmieren Sie die entsprechende **decompress** Funktion, die das Gegenteil macht.

c) Analysieren Sie die Komplexität der Funktionen von a) und b).

2. Aufgabe (9 Punkte)

a) Programmieren Sie folgende Funktionen für den algebraischen Datentyp **ZInt** aus der Vorlesung.

```
zneg  :: ZInt -> ZInt                -- negative Zahl
zabs  :: ZInt -> ZInt                -- absoluter Wert
zpow  :: ZInt -> Nat  -> ZInt        -- Potenz Funktion
zIsDivisor :: ZInt -> ZInt -> Bool   — Teilerfunktion
zggt :: ZInt -> ZInt -> ZInt         -- größter gemeinsamer Teiler
```

b) Definieren Sie folgende Hilfsfunktionen, um das Testen zu vereinfachen.

```
zint2Int :: ZInt    -> Integer       -- transformiert von ZInt nach Integer
int2Zint :: Integer -> ZInt          -- transformiert von Integer nach ZInt
```

3. Aufgabe (10 Punkte)

Definieren Sie für folgende algebraische Datentypdefinition (Baumstruktur ohne Inhalt aus der Vorlesung)

```
data SimpleBT = L | N SimpleBT SimpleBT     deriving Show
```

folgende Funktionen:

```
insLeaf   :: SimpleBT -> SimpleBT              -- ein Blatt wird in den Baum eingefügt
insLeaves :: SimpleBT -> Integer -> SimpleBT   -- eine eingegebene Anzahl von Blättern
```

```
                                                -- wird in den Baum eingefügt
delLeaf :: SimpleBT -> SimpleBT                 -- löscht ein Blatt aus dem Baum
delLeaves :: SimpleBT -> Integer -> SimpleBT    -- eine eingegebene Anzahl von Blättern
                                                -- wird in dem Baum gelöscht
```

Alle Funktionen sollen selber dafür sorgen, dass der Baum möglichst balanciert bleibt.

4. Aufgabe (8 Punkte)

Erweitern Sie die Funktionen für den algebraischen Datentyp **BSearchTree** (Binäre Suchbäume) um folgende Funktionen:

```
twoChildren :: (Ord a) => BSearchTree a -> Bool
    -- entscheidet, ob jeder Knoten des Baums genau zwei Kinder hat oder nicht
full :: (Ord a) => BSearchTree a -> Bool
    -- überprüft, ob ein Baum vollständig ist.
mapTree :: (Ord a, Ord b) => (a -> b) -> BSearchTree a -> BSearchTree b
foldTree :: (Ord a) => b -> (a -> b -> b -> b) -> BSearchTree a -> b
```

Wichtige Hinweise:

1) Zu jeder Funktion sollen Testfunktionen geschrieben werden.

2) Verwenden Sie geeignete Namen für Ihre Variablen und Funktionsnamen, die den semantischen Inhalt der Variablen oder die Semantik der Funktionen wiedergeben.

3) Verwenden Sie vorgegebene Funktionsnamen, falls diese angegeben werden.

4) Kommentieren Sie Ihre Programme.

5) Verwenden Sie geeignete lokale Funktionen und Hilfsfunktionen in Ihren Funktionsdefinitionen.

5) Schreiben Sie in allen Funktionen die entsprechende Signatur.