PHILIPPS-UNIVERSITÄT MARBURG
Fachbereich Mathematik und Informatik
Prof. Dr. R. Loogen

D-35032 Marburg
Hans Meerwein Straße
Lahnberge

12. November 2003

Übungen zur „Praktischen Informatik III“, WS 2003/04

Nr. 4, Besprechung bzw. Abgabe: 19. und 20. November in den Übungen

---

Importieren Sie das auf der Vorlesungsseite zur Verfügung gestellte Modul `SearchTree.hs` durch die Angabe von `import SearchTree` am Anfang Ihres Programms. Der folgende Auszug aus der Spezifikation zeigt, welche Operationen aus dem Modul exportiert werden. Nur diese können in den Aufgaben zur Baumverarbeitung verwendet werden.

```
module SearchTree
  (STree,
   nil,       -- STree a
   node,      -- a -> STree a -> STree a -> STree a
   isNil,     -- STree a -> Bool
   isNode,    -- STree a -> Bool
   leftSub,   -- STree a -> STree a
   rightSub,  -- STree a -> STree a
   rootVal,   -- STree a -> a
   insTree,   -- Ord a => a -> STree a -> STree a
   delete,    -- Ord a => a -> STree a -> STree a
   minTree,   -- Ord a => STree a -> a
   showTree   -- STree a -> IO ()
  )
where ...
```

A. Mündliche Aufgaben

17. Sortieren mit Suchbäumen

    (a) Schreiben Sie eine Funktion `list2Tree :: Ord a => [a] -> STree a` zur Überführung einer Liste von Werten, auf denen eine Ordnungsrelation besteht (Kontext `Ord a =>`), in einen Suchbaum.

    (b) Schreiben Sie eine Funktion `tree2OrderedList :: Ord a => STree a -> [a]` die die in einem Suchbaum gespeicherten Werte als geordnete Liste ausgibt und dazu die Funktion `minTree` verwendet.

    Diese beiden Funktionen liefern durch Hintereinanderausführung ein Sortierverfahren:

    ```
    treeSort    :: Ord a => [a] -> [a]
    treeSort xs =  tree2OrderedList (list2Tree xs)
    ```

18. Bäume mit Größenangabe

    Zur effizienteren Bestimmung der Größe von Suchbäumen soll die Implementierung des `searchTree`-Moduls so abgeändert werden, dass in den Knoten neben dem Wert auch die Größe des entsprechenden Teilbaums gespeichert wird. Zur Implementierung von Suchbäumen soll die folgende Datenstruktur eingesetzt werden:

```
data STree a = Nil | Node a Int (STree a) (STree a)
```

Passen Sie die Implementierung des Moduls `SearchTree` an die geänderte Datenstrukturdeklaration an.

B. Hausaufgaben

Hinweise: Die Lösungen sollten grundsätzlich schriftlich, Programme zusätzlich auf Diskette oder per E-Mail an Ihren Tutor abgegeben werden. Die Abgabe ist in Gruppen bis zu zwei Personen erlaubt.

19. Sparschwein — 2 Punkte

Implementieren Sie einen abstrakten Datentyp Sparschwein mit den Operationen:

```
emptyPig :: PiggyBank                          -- leeres Sparschwein
add      :: Coin -> PiggyBank -> PiggyBank     -- Einwerfen einer Muenze
shake    :: PiggyBank -> (PiggyBank, Coin)     -- Herausschtteln einer Muenze
isEmpty  :: PiggyBank -> Bool                  -- Test auf leeres Sparschwein
break    :: PiggyBank -> Money                 -- Aufbrechen des Sparschweins
```

Dabei seien die Datentypen `Coin` und `Money` wie folgt definiert:

```
data Coin  = OneCent | TwoCents | FiveCents | TenCents | FiftyCents
           | OneEuro | TwoEuros
type Money = Int -- Geldwert in Cents
```

20. Baum-Balancierung — 6 Punkte

Ein binärer Baum heißt *balanciert*, wenn sich die Anzahl der Knoten im linken und rechten Teilbaum um höchstens eins unterscheiden und wenn beide Teilbäume balanciert sind. Der leere Baum ist per definitionem balanciert.

(a) Definieren Sie eine Funktion `size :: STree a -> Int` zur Bestimmung der Anzahl der Knoten in einem Suchbaum. / 1

(b) Schreiben Sie eine Funktion `isBalanced :: STree a -> Bool` die testet, ob ein gegebenener Baum balanciert ist. / 2

(c) Entwickeln Sie eine Funktion `balance :: Ord a => STree a -> STree a` die einen beliebigen Suchbaum in einen balancierten Suchbaum mit denselben Einträgen umwandelt. / 3

21. Holländisches Flaggenproblem — 4 Punkte

Gegeben sei die folgende Typdeklaration zur Definition gefärbter Objekte:

```
data ColObject a = Red a | White a | Blue a
```

Schreiben Sie eine Funktion `rwb :: [ColObject a] -> [ColObject a]`
die eine Liste gefärbter Objekte so umordnet, daß zuerst die roten Objekte, anschliessend alle weißen Objekte und zuletzt alle blauen Objekte auftreten. Die relative Ordnung gleichfarbiger Elemente soll dabei erhalten bleiben.