TU Dresden
Fakultät Informatik
Institut für Theoretische Informatik
Lehrstuhl Grundlagen der Programmierung
Prof. Dr.-Ing. H. Vogler / Dr.-Ing. L. Rüdiger

SS 2011

2. Übung

Fakultät Verkehrswissenschaften
Fachrichtung Verkehrsingenieurwesen

Zeitraum: 02. bis 12.05.2011

Aufgabe 4: Einfache Funktionen in HASKELL

Programmieren Sie eine Funktion f in Haskell, die eine Liste von Int–Zahlen als Eingabe nimmt und eine Liste von Int–Zahlen liefert, welche nur die Zahlen der Ausgangsliste, die größer als Null sind, in gleicher Reihenfolge enthält.

Aufgabe 5: Algebraische Datentypen in HASKELL

Gegeben sei die folgende Typvereinbarung für binäre Bäume:

```
data Tree = Leaf Int | Branch Tree Tree
```

(a) Schreiben Sie einen Baum dieses Typs mit mindestens 5 Blattknoten auf.

(b) Schreiben Sie folgende Funktionen für o.g. Datentyp auf:

1. Zum Ermitteln der Anzahl der Blattknoten.
2. Zum Ermitteln der Summe aller Blattinformationen .

Aufgabe 6: Polymorphe Datentypen in HASKELL

Gegeben sei der folgende polymorphe algebraische Datentyp:

```
data Tree a = Branch (Tree a) (Tree a) | Leaf a
```

mit dessen Hilfe sich binäre Bäume konstruieren lassen, die an den Blättern Werte des Typs **a** speichern.

Programmieren Sie eine Funktion `check :: Tree Bool -> Bool`, die genau dann **True** liefert, wenn der Eingabebaum mindestens ein Blatt mit dem gespeicherten Wert **False** besitzt.

Aufgabe 7: Funktionen in HASKELL

Gegeben sei der folgende Datentyp **Tree** zur Realisierung von Binärbäumen mit Knotenbeschriftungen vom Typ **Int** in Haskell:

```
data Tree = Node Int Tree Tree | Nil
```

Definieren Sie unter Verwendung der Haskell–Funktion `collapse`

```
collapse :: Tree -> [Int]
collapse Nil = []
collapse (Node x y z) = (collapse y) ++ [x] ++ (collapse z)
```

eine Haskell–Funktion `check :: Tree -> Bool`, die überprüft, ob es sich bei einem Baum vom Typ `Tree` um einen binären Suchbaum handelt, d. h. jeder Knoten ist mit einer ganzen Zahl beschriftet und es muss für jeden Knoten **x** gelten, dass seine Beschriftung größer oder gleich (bzw. kleiner oder gleich) allen Beschriftungen im linken (bzw. rechten) Teilbaum von **x** ist. Sie können die übliche Relation $\leq$ auf ganzen Zahlen benutzen.

Aufgabe 8: Funktionen in HASKELL

Definieren Sie eine polymorphe Haskell–Funktion

```
f :: [a] -> [a]
```

die aus einer Liste von Elementen des Typs **a** jedes zweite Element entfernt.
Hinweis: Überlegen Sie sich zunächst eine Strategie zur Lösung dieser Aufgabenstellung. Schreiben Sie gegebenenfalls eine Hilfsfunktion.

Aufgabe 9: Auswertung von Funktionen in HASKELL

Eine Liste $\mathbf{s} = [e_1, e_2, \ldots]$ bestehe aus Funktionswerten $f(i) = e_i$ mit $i \in I\!N^+$.
f sei wie folgt definiert:

$f(1) = 1, f(2) = 1, f(3) = 1$
$f(i+3) = f(i) + f(i+1) + f(i+2)$ für $(i \geq 1)$

(a) Geben Sie eine Funktion **fl** an, die die oben definierte Liste erzeugt.

(b) Schreiben Sie eine Funktion **p**, die von der Liste **s** das n-te Element ermittelt.

Aufgabe 10: Unifikationsalgorithmus

Es sei δ ein dreistelliges, σ ein zweistelliges, γ ein einstelliges und α ein nullstelliges Basisfunktionssymbol. Des Weiteren sei $V = \{x_1, \ldots, x_3\}$ eine Menge von Variablen.

Wenden Sie den Unifikationsalgorithmus auf die Terme t_1 und t_2 an und ermitteln Sie den allgemeinsten Unifikator:

$t_1 = \delta(\gamma(x_1), \gamma(\gamma(\alpha)), \sigma(x_2, \alpha))$

$t_2 = \delta(\gamma(\alpha), \gamma(x_2), x_3)$