Prof. Dr. Uwe Schmidt

fhwedel
UNIVERSITY OF APPLIED SCIENCES

20. August 2012

---

Aufgaben zur Klausur **Grundlagen der funktionalen Programmierung** im SS 2012 (BInf 13b)

Zeit: 90 Minuten erlaubte Hilfsmittel: keine

Bitte tragen Sie Ihre Antworten und fertigen Lösungen ausschließlich an den gekennzeichneten Stellen in das Aufgabenblatt ein. Ist ihre Lösung wesentlich umfangreicher, so überprüfen Sie bitte nochmals Ihren Lösungsweg.

Sollten Unklarheiten oder Mehrdeutigkeiten bei der Aufgabenstellung auftreten, so notieren Sie bitte, wie Sie die Aufgabe interpretiert haben.

Viel Erfolg!

Diese Klausur besteht einschließlich dieses Deckblattes aus 6 Seiten.

---

**Aufgabe 1:**

Die `filter`-Funktion dient zum Löschen aller Elemente einer Liste, die eine bestimmte Eigenschaft nicht erfüllen. Sie besitzt folgenden Typ:

$$\mathit{filter} :: (a \to \mathit{Bool}) \to [a] \to [a]$$

Definieren Sie `filter` mit Hilfe einer Listcomprehension

. . . . . .

. . . . . .

. . . . . .

Definieren Sie `filter` mit Hilfe einer rekursiven Funktion

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

---

**Aufgabe 2:**

Gegeben seien die folgenden rekursiven Funktionen. Diese besitzen eine ähnliche Struktur.

$$
\begin{array}{ll}
sum & :: Num\ a \Rightarrow [a] \to a \\
sum\ xs & = sum'\ 0\ xs \\
\quad \textbf{where} & \\
\quad sum'\ r\ [\,] & = r \\
\quad sum'\ r\ (x : xs) & = sum'\ (r + x)\ xs \\
reverse & :: [a] \to [a] \\
reverse\ xs & = reverse'\ [\,]\ xs \\
\quad \textbf{where} & \\
\quad reverse'\ r\ [\,] & = r \\
\quad reverse'\ r\ (x : xs) & = reverse'\ (x : r)\ xs
\end{array}
$$

Um solche ähnlich strukturierten Funktionen einfacher zu implementieren, gibt es die so genannten `fold`-Funktionen. `foldl` ist eine dieser Funktionen mit folgendem Typ:

$$foldl :: (b \to a \to b) \to b \to [a] \to b$$

Implementieren Sie diese `foldl`-Funktion:

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Reimplementieren sie `sum` mit Hilfe von `foldl`

. . . . . .

. . . . . .

Reimplementieren sie `reverse` mit Hilfe von `foldl`

. . . . . .

. . . . . .

. . . . . .

**Aufgabe 3:**

Gegeben sei ein Datentyp `Interval` für abgeschlossene Intervalle der ganzen Zahlen.

$$\textbf{type}\ \mathit{Interval} = (\mathit{Int}, \mathit{Int})$$

Ein Wert `(i,j)` aus diesem Wertebereich soll dabei die Menge der Zahlen $\{i, i+1, ..., j\}$ repräsentieren. Entwickeln Sie eine Funktion `overlap`, mit der getestet werden kann, ob zwei Intervalle gemeinsame Elemente haben oder aneinander stoßen.

Hier sind einige Testfälle gegeben:

$\mathit{overlap}\ (1,2)\ (4,5) \equiv \mathit{False}$
$\mathit{overlap}\ (1,2)\ (0,5) \equiv \mathit{True}$
$\mathit{overlap}\ (1,2)\ (3,3) \equiv \mathit{True}$
$\mathit{overlap}\ (1,1)\ (3,3) \equiv \mathit{False}$

$\mathit{overlap} :: \mathit{Interval} \rightarrow \mathit{Interval} \rightarrow \mathit{Bool}$

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Entwickeln Sie weiter eine Funktion `merge`, mit der das kleinste Intervall berechnet wird, das zwei gegebene Intervalle enthält.

$$merge :: Interval \to Interval \to Interval$$

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

---

**Aufgabe 4:**

Es sei folgender rekursiver Datentyp gegeben:

$$\textbf{data}\ \mathit{Tree}\ a = \mathit{Leaf}\ \ a \mid \mathit{Node}\ (\mathit{Tree}\ a)\ (\mathit{Tree}\ a)$$

Entwickeln Sie eine Funktion `mapTree`, mit der ein Baum in einen strukturgleichen Baum transformiert wird, indem alle Elemente mit einer Funktion verarbeitete werden.

$$\mathit{mapTree} :: (a \to b) \to \mathit{Tree}\ a \to \mathit{Tree}\ b$$

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Gegeben sei ein Beispiel-Baum mit Zahlen als Elementen

$$\textbf{type}\ \mathit{TreeOfInt} = \mathit{Tree}\ \mathit{Int}$$
$$\mathit{t1} :: \mathit{TreeOfInt}$$
$$\mathit{t1} = \mathit{Node}\ (\mathit{Leaf}\ 1)\ (\mathit{Node}\ (\mathit{Leaf}\ 3)\ (\mathit{Leaf}\ 4))$$

Entwickeln Sie eine Funktion, die solche `TreeOfInt`-Bäume verarbeitet, indem alle Werte an den Blättern um 1 erhöht werden.

. . . . . .

. . . . . .

Entwickeln Sie eine Funktion `sumTree`, die die Elemente in einem `TreeOfInt`-Baum aufsummiert.

. . . . . .

. . . . . .

. . . . . .

. . . . . .