10. Aufgabenblatt zu Funktionale Programmierung vom 17.12.2004, fällig: 12.01.2005

Themen: *Funktionen auf ganzen Zahlen, Funktionale auf Bäumen*

Für das letzte Aufgabenblatt in diesem Semester sollen Sie wie schon für die Aufgabenblätter 2 und 6 ein **"literate"** Haskell-Script schreiben. Die im einzelnen zu entwickelnden Haskell-Rechenvorschriften zur Lösung der unten angegebenen Aufgabenstellungen sollen Sie dann für die Abgabe in einer Datei namens `Aufgabe10.lhs` ablegen. Versehen Sie wie auf den bisherigen Aufgabenblättern alle Funktionen, die Sie zur Lösung brauchen, mit ihren Typdeklarationen und kommentieren Sie Ihre Programme aussagekräftig. Benutzen Sie, wo sinnvoll, Hilfsfunktionen und Konstanten. Im einzelnen sollen Sie folgende Problemstellungen bearbeiten:

1. In dieser Aufgabe wollen wir in Analogie zu den in Haskell auf Listen vordefinierten Funktionalen `map`, `filter` und `fold` entsprechende Funktionale für die auf Aufgabenblatt 9 eingeführten polymorphen Bäume schreiben.

```
data Tree a = Leaf Int a |
              Node Int a (Tree a) (Tree a)
```

   (a) Schreiben Sie ein Funktional `mapTree` mit der Signatur `mapTree :: (a -> b) -> Tree a -> Tree b`. Ähnlich wie `map` die Elemente von Listen mittels der als Argument übergebenen Funktion transformiert, so transformiert `mapTree` auf Bäumen die Marken dieser Bäume gemäß der Abbildungsvorschrift der dem Funktional `mapTree` als Argument übergebenen Funktion.

   (b) Schreiben Sie ein Funktional `filterTree` mit der Signatur `filterTree :: (a -> Bool) -> String -> Tree a -> [a]`. Angewendet auf ein Prädikat, eine Zeichenreihe mit den Werten `Praefix`, `Postfix` oder `Infix` und einen Baum liefert das Funktional `filterTree` als Resultat eine Liste all derjenigen Knoten- und Blattmarkierungen, die das als erstes Argument übergebene Prädikat erfüllen. Der Wert des zweiten Arguments legt dabei fest, in welcher Reihenfolge die Knoten des Baumes besucht werden und dadurch auch in welcher Reihenfolge die Marken in der Ergebnisliste aufscheinen. Die erste auszugebende Marke steht an erster Position in der Ergebnisliste, die zweite an der zweiten usw., d.h. neu als auszugeben gefundene Marken werden stets hinten an die aktuelle Resultatliste angefügt. Ist das Zeichenreihenargument von

`Praefix`, `Postfix` und `Infix` verschieden, so soll mithilfe von `error` die Programmausführung mit der Meldung `Fehlerhafte Eingabe` abgebrochen werden.

(c) Schreiben Sie ein Funktional `foldTree` mit der Signatur `foldTree :: (a -> b -> b) -> b -> String -> Tree a -> b`. Durch das erste Argument ist dabei die Faltungsvorschrift festgelegt, durch das zweite Argument der Startwert und durch das dritte Argument die Durchlaufordnung durch den Baum. Dieses dritte Argument wird in genau derselben Weise (einschließlich Fehlerbehandlung) wie das entsprechende Argument des Funktionals `filterTree` verwendet, d.h. insbesondere, dass mithilfe von `error` die Programmausführung mit der Meldung `Fehlerhafte Eingabe` abgebrochen werden soll, wenn das dritte Argument einen von `Praefix`, `Postfix` und `Infix` verschiedenen Wert hat.

2. *Weihnachtsaufgabe*: Fühlen Sie sich fit für einen Job bei *Google*? Im Oktober diesen Jahres forderte *Google* potenzielle Dienstnehmer zur Bewerbung auf. Erfolgreich konnten allerdings nur Bewerbungen sein, denen die Lösung einer Reihe von Aufgaben beigelegt war. Eine dieser Aufgaben lautete im Wortlaut wie folgt:

```
Consider a function which, for a given whole number n,
returns the number of ones required when writing out
all number between 0 and n. For example, f(13)=6. Notice
that f(1)=1. What is the next largest n such that f(n)=n?
```

In der Folge nennen wir eine Zahl n mit der Eigenschaft $f(n) = n$ eine Google-Zahl. Ihre Aufgabe ist nun die folgende:

(a) Schreiben Sie eine Haskell-Rechenvorschrift `f` mit der Signatur `f :: Integer -> Integer`, die das in *Googles* Aufgabe beschriebene Abbildungsverhalten hat.

(b) Schreiben Sie eine Haskell-Rechenvorschrift `fnext` mit der Signatur `fnext :: Integer -> Integer`. Wird `fnext` mit einem negativen Argument m aufgerufen, so ist das Resultat die Zahl 0, ansonsten ist das Resultat die kleinste ganze positive Zahl n mit $n > m$ und `f n = n`.

(c) Schreiben Sie eine Haskell-Rechenvorschrift `fmore` mit der Signatur `fmore :: Integer -> [Integer]`. Aufgerufen mit dem Argument m, liefert `fmore` aufsteigend in einer Liste geordnet die m betragskleinsten positiven ganzen Zahlen `n` mit der Eigenschaft `f n = n` zurück. Gibt es weniger als m Zahlen mit dieser Eigenschaft, dann so viele als möglich. Beispiel: Der Aufruf `fmore 1` liefert als Ergebnis die Liste `[1]`, der Aufruf `fmore 2` die Liste `[1,fnext 1]`. Der Aufruf `last . fmore 2` liefert also das Resultat der Funktion `fnext` angewendet auf das Argument 1.

Für die erste Teilaufgabe der Weihnachtsaufgabe geht es wie üblich darum, ob pro Testfall 10 Punkte abgezogen werden oder nicht. Anhand der zweiten Teilaufgabe können Sie überprüfen, ob für Sie eine Bewerbung bei Google in nächster Zukunft in Frage kommt. Für die dritte Teilaufgabe wird ein Schokoladenweihnachtsmann aus 2004 oder wahlweise eine Flasche Sekt als Preis ausgelobt. Gewinnen wird diesen Schokoladenweihnachtsmann bzw. Flasche Sekt, wer die meisten Google-Zahlen in beschränkter Zeit findet. Finden Sie bei Ihren Überlegungen heraus, dass es zu jeder vorgegebenen Zahl eine mindestens so große Google-Zahl gibt, so fügen Sie Ihrer Resultatliste am Ende eine -1 an und fügen Ihrem Haskell-Skript einen Kommentar, ggf. auch eine Berechnungsfunktion bei, aus dem hervorgeht, wie Sie zu Ihrem Schluss kommen. Bei mehreren gleichwertigen Lösungen entscheidet das Los. Wie bei Gewinnspielen üblich, kann Korrespondenz darüber nicht geführt werden und ist der Rechtsweg ausgeschlossen. Die Preisverleihung findet im Januar in einer der Vorlesungen statt.

Interesse gefunden bei *Google* zu arbeiten? Die vollständige Bewerbungsaufforderung mit allen Fragen finden Sie in der Oktoberausgabe 2004 der Communications of the ACM (CACM 47 (10), 2004).

Frohe Weihnachten,
einen guten Rutsch und
viel Erfolg im neuen Jahr!