**6. Übungsaufgabe zu Funktionale Programmierung**
**Themen: Typklassen und ihre Instanzen**
**ausgegeben: 03.12.2003, fällig: 10.12.2003**

Für dieses Aufgabenblatt sollen Sie Haskell-Rechenvorschriften zur Lösung der im folgenden angegebenen Aufgabenstellungen entwickeln und für die Abgabe in einer Datei namens `Aufgabe6.hs` in Ihrem Homeverzeichnis ablegen. Kommentieren Sie Ihre Programme wieder aussagekräftig und benutzen Sie, wo sinnvoll, Hilfsfunktionen und Konstanten.

In den folgenden Aufgaben betrachten wir den wie folgt definierten polymorphen Datentyp `PBinTree` zur Darstellung binärer Bäume.

```
data PBinTree a = Empty |
                  Node a (PBinTree) (PBinTree a)
```

- Schreiben Sie eine Haskell-Rechenvorschrift `tiefe`, die die Tiefe von Bäumen des Typs `PBinTree` zurückliefert. Dabei gelte, dass der leere Baum die Tiefe 0 habe, ansonsten die Tiefe eines Baumes durch die Anzahl der Knoten auf dem längsten Pfad von der Wurzel zu einem Blatt gegeben ist.

- Binäre Bäume vom Typ `PBinTree` heißen *vollständig*, wenn alle ihre Knoten entweder keinen oder zwei von `Empty` verschiedene Nachfolger haben (d.h. es gibt keinen Knoten, dessen zwei Nachfolger von der Form `Empty` und `Node a (PBinTree a) (PBinTree a)` sind. Sie heißen *perfekt balanciert*, wenn alle Pfade von der Wurzel zu einem Blatt dieselbe Länge besitzen, und *balanciert*, wenn sich die Länge aller Pfade von der Wurzel zu einem Blatt höchstens um 1 unterscheidet. Die Länge eines Pfades ist dabei durch die Zahl der Knoten auf dem Pfad gegeben.

  Schreiben Sie Haskell-Rechenvorschriften `istVollstaendig`, `istBalanciert` und `istPerfektBalanciert`, die testen, ob ein vorgelegter binärer Baum vollständig, balanciert oder sogar perfekt balanciert ist und einen entsprechenden Wahrheitswert ausgeben.

- Schreiben Sie Haskell-Rechenvorschriften `praeOrder`, `inOrder` und `postOrder`, die einen binären Baum in Präfix-, Infix- und Postfix-Ordnung durchlaufen und die dabei aufgefundenen Marken in der Reihenfolge ihres Auffindens in einer Liste ausgeben, d.h. die Marke des i-ten besuchten Knotens ist das i-te Element in der Liste.

*Hinweis*: In Präfixordnung wird zunächst die Marke an der Wurzel ausgegeben, anschließend werden die Marken des linken Teilbaums entsprechend der Präfixordnung, danach die Marken des rechten Teilbaums in Präfixordnung ausgegeben. In Infixordnung werden zunächst die Marken des linken Teilbaums in Infixordnung ausgegeben, danach die Marke der Wurzel, zum Schluss die Marken des rechten Teilbaums in Infixordnung. In Postfixordnung werden zunächst die Marken des linken Teilbaums in Postfixordnung ausgegeben, danach die Marken des rechten Teilbaums in Postfixordnung, zum Schluss die der Wurzel.

- In dieser Aufgabe betrachten wir eine Instanz der eingangs eingeführten binären Bäume; die Instanz über Zeichenreihen. Schreiben Sie eine Haskell-Rechenvorschrift `protokoll`, die angewendet auf einen binären Baum über Zeichenreihen diesen in Infixordnung durchläuft und die dabei gefundenen Marken konkateniert. Marken neu besuchter Knoten werden dabei stets vorne als Präfix angefügt. Die so entstehende Gesamtzeichenreihe wird dann schliesslich als Resultat ausgegeben.