**3. Übungsaufgabe zu**
**Fortgeschrittene Funktionale Programmierung**
**Themen: Pretty Printing und Typanalyse**
**ausgegeben: 03.06.2004, fällig: 16.06.2004**

Für dieses Aufgabenblatt sollen Sie Haskell-Rechenvorschriften zur Lösung der im folgenden angegebenen Aufgabenstellungen entwickeln und für die Abgabe in einer Datei namens `AufgabeFFP3.hs` in Ihrem Homeverzeichnis ablegen, wie gewohnt auf oberstem Niveau. Kommentieren Sie Ihre Programme aussagekräftig und benutzen Sie, wo sinnvoll, Hilfsfunktionen und Konstanten.

- In dieser Aufgabe betrachten wir die Syntaxbäume von Programmen der Programmiersprache **While**, wie sie von Ihrem Programm aus Aufgabenblatt 2 erzeugt werden. Nach dem Muster der Vorlesung sollen nun "Pretty Printer" (Schöndrucker) zur Ausgabe dieser Syntaxbäume (die dem Schöndrucker etwa als Resultat der Syntaxanalyse oder auch direkt übergeben werden können) geschrieben werden.
  - Schreiben Sie zunächst einen Schöndrucker, in dem die Ausgabe von Teilbäumen eines Baums nach dem auf Folie 8 aus Vorlesungsteil 5 dargestellten Layout-Muster erfolgt. Was entspricht in Ihrer Datenstruktur von Syntaxbäumen dem Konstruktor `Node` und was der Zeichenreihe, mit der `Node` annotiert ist? Überlegen Sie sich ggf., was Sie bei eventuell fehlender Entsprechung zur `Node` annotierenden Zeichenreihe sinngemäß ausgeben können.
  - Ändern Sie den Schöndrucker aus dem vorigen Aufgabenteil so ab, dass die Ausgabe von Bäumen dem Layout-Muster von Folie 10 aus Vorlesungsteil 5 folgt.
  - Erweitern Sie die Schöndrucker aus den vorigen beiden Aufgabenteilen so, dass eine vom Anwender vorgebene max. Zeilenlänge berücksichtigt wird. Bei mehren möglichen Layouts erfolge die Wahl eines "besten" Layouts dabei nach den auf Folie 31 aus Vorlesungsteil 5 gegebenen Kriterien.
  - Testen Sie die verschiedenen Schöndrucker mit einigen eigenen aussagekräftigen Beispielen. Geben Sie diese Beispiele mit an (ggf. in Form eines Kommentars).

- In dieser Aufgabe betrachten wir einige Typanalyseprobleme über Ausdrücken. Dazu seien die folgenden Haskell-Ausdrücke gegeben, wobei `curry` und `uncurry` die bekannten Funktionen zur Curryfikation und Decurryfikation und `id` die Identitätsfunktion bezeichnen:

  ```
  curry id
  uncurry id
  uncurry curry
  curry uncurry
  ```

  Überlegen Sie sich, welche der obigen Ausdrücke sich in Haskell typen lassen und welche nicht und geben Sie jeweils eine kurze Begründung dafür an. Im positiven Fall auch, wie der (allgemeinste) Typ aussieht. Sie brauchen zur Lösung dieser Aufgabe nichts zu programmieren, Sie können Ihre Vermutung aber mithilfe von Hugs überprüfen und anschließend Ihre begründete Antwort zu dieser Aufgabe in Form eines Kommentars in der Datei `AufgabeFFP3.hs` mit abgeben.