PHILIPPS-UNIVERSITÄT MARBURG
Fachbereich Mathematik und Informatik
Prof. Dr. R. Loogen

D-35032 Marburg
Hans Meerwein Straße
Lahnberge

22. Oktober 2003

Übungen zur „Praktischen Informatik III“, WS 2003/04

Nr. 1, Besprechung bzw. Abgabe: 29. und 30. Oktober in den Übungen

---

A. Mündliche Aufgaben

1. Vergleich von Argumenten

   (a) Geben Sie Definitionen der folgenden Funktionen an:

   ```
   howManyEqual      :: Int -> Int -> Int -> Int
   howManyOfTwoEqual :: Int -> Int -> Int
   allDifferent      :: Int -> Int -> Int -> Bool
   ```

   `howManyEqual` und `howManyOfTwoEqual` zählen, wieviele der gegebenen Funktionsargumente gleich sind. `allDifferent` liefert den Wert `True` zurück, wenn alle drei Argumente verschieden sind.

   Der Operator `/=` liefert beim Aufruf `m /= n` den Wert `True`, wenn `m` ungleich `n` gilt.

   (b) Geben Sie Testdaten an, die zeigen, dass die folgende alternative Definition von `allEqual` falsch ist. Wo liegt das Problem?

   ```
   allEqual'       :: Int -> Int -> Int -> Bool
   allEqual' n m p =  ((n+m+p) == 3*p)
   ```

2. Reduktion

   (a) Bestimmen Sie alle Redexe im folgenden Ausdruck:

   ```
   allEqual (simple (2-1) (square (2+1)) (square (2+2)))
            (square (3+2)) (5*(3+2))
   ```

   (b) Führen Sie eine ausführliche Auswertung des Ausdrucks durch.

3. Teiler

   Schreiben Sie eine Funktion `anzahlTeiler :: Int -> Int`, die zu einer gegebenen Zahl `n >= 1`, die Anzahl `k` der Zahlen `m` mit `1 <= m <= n` bestimmt, die `n` teilen.

B. Hausaufgaben

Hinweise: Die Lösungen sollten grundsätzlich schriftlich, Programme zusätzlich auf Diskette oder per E-Mail an Ihren Tutor abgegeben werden. Die Abgabe ist in Gruppen bis zu zwei Personen erlaubt.

4. Bildtransformationen — 7 Punkte

   Das Modul `Picture` in der auf der Vorlesungsseite bereitgestellten Datei `picture.hs` stellt folgende Funktionen für einfache Bildtransformationen zur Verfügung:

```
module Picture where
type Picture = ...

-- Bildschirmausgabe von Bildern
printPic :: Picture -> IO()

-- Spiegelung horizontal, vertikal und diagonal
flipH :: Picture -> Picture
flipV :: Picture -> Picture
flipD :: Picture -> Picture

-- Farbinversion
invertColour :: Picture -> Picture

-- Anordnung von Bildern uebereinander bzw. nebeneinander
above :: Picture -> Picture -> Picture
aside :: Picture -> Picture -> Picture

-- vorgegebene Bilder
white  :: Picture  -- weisses Rechteck
lambda :: Picture  -- Lambda
```

Durch Angabe von `import Picture` am Beginn Ihres Programms können Sie das Modul importieren und die Funktionen verwenden.

(a) Geben Sie zwei verschiedene Möglichkeiten an, ein Bild der folgenden Gestalt zu erzeugen: / 2

```
......######
......######
......######
######......
######......
######......
```

(b) Definieren Sie eine Funktion `rotate90 :: Picture -> Picture`, die ein Bild um 90 Grad nach rechts dreht. / 2

(c) Definieren Sie eine Funktion / 3

```
chessboard :: Int -> Picture -> Picture
```

die zu einer positiven Zahl `n` und einem Bild ein Schachbrett erzeugt, in dem das Bild in jeder Dimension `n`-mal abwechselnd positiv und negativ, d.h. mit invertierten Farben, wiederholt wird. `chessboard 8 white` definiert also ein normales Schachbrett.

5. Perfekte Zahlen **5 Punkte**

Eine natürliche Zahl heißt *perfekt*, wenn sie der Summe ihrer (echten) Teiler entspricht. Zum Beispiel ist die Zahl 6 perfekt, da $6 = 1 + 2 + 3$. Die Zahl 8 ist hingegen nicht perfekt, da $8 \neq 1 + 2 + 4 = 7$.

Schreiben Sie eine Testfunktion `perfekt :: Int -> Bool`, die zu einer gegebenen Zahl feststellt, ob sie perfekt ist.