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

D-35032 Marburg
Hans Meerwein Straße
Lahnberge

26. November 2003

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

Nr. 6, Besprechung bzw. Abgabe: 3. und 4. Dezember in den Übungen

---

A. Mündliche Aufgaben

26. Interaktive Ein-/Ausgabe

Schreiben Sie zwei interaktive Programme, die das folgende wiederholt tun, bis die Eingabe einer Leerzeile erfolgt:

(a) Eine Eingabezeile in Großbuchstaben wieder ausgeben.
Die vordefinierte Funktion `toUpper :: Char -> Char` wandelt Klein- in Großbuchstaben um.

(b) Klammern in einer Eingabezeile überprüfen:
Als Klammern sollen geschweifte (`"{}"`), runde (`"()"`) und eckige (`"[]"`) Klammern zugelassen sein. Das Programm soll testen, ob von diesen Klammern jeweils genausoviele öffnende wie schließende in einer korrekten Reihenfolge in der Eingabezeile enthalten sind. `{()[()()]}` ist zum Beispiel korrekt, `([)` aber nicht.

27. Interaktive Ein-/Ausgabe fuer Countdown-Programm

Erweitern Sie das in Aufgabe 25 entwickelte Programm für das Countdown-Problem um eine Ein-/Ausgabefunktion, die nach entsprechenden Eingabeaufforderungen die Zahlenliste und die Zielzahl einliest und anschließend eine erste Lösung des Countdown-Problems berechnet und ausgibt. Weitere Lösungen sollen nur nach Eingabe von `'\n'` (Returntaste) ausgegeben werden. Bei Betätigung einer anderen Eingabetaste oder, falls keine weiteren Lösungen existieren, soll das Programm stoppen.

B. Hausaufgaben

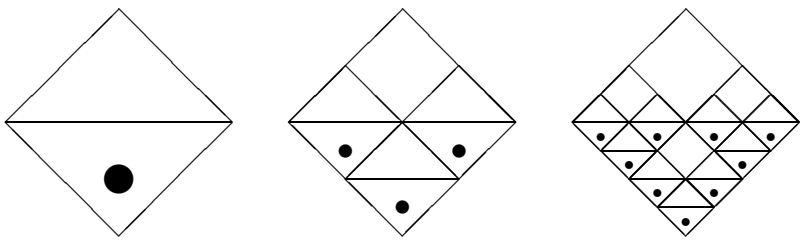
Hinweis: Für die Hausaufgaben wird die *Hugs Graphics Library* benötigt, die von der Internet-Seite `http://haskell.org/graphics/` geladen werden kann. Das Importieren der Bibliothek erfolgt durch `import SOEGraphics`.

---

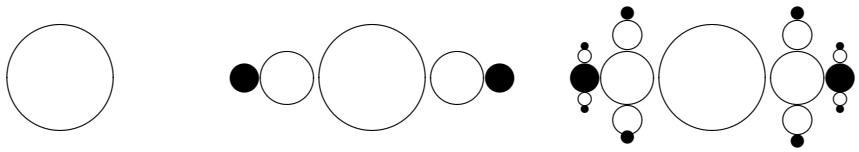
28. Fraktale Bilderzeugung — 8 Punkte

Schreiben Sie Programme, die jeweils ein fraktales Bild gemäß den folgenden Konstruktionsvorschriften erzeugen und in einem Fenster ausgeben:

(a) Vierecke — / 4



(b) Kreise / 4



Das Bild soll in der skizzierten Weise abwechselnd horizontal und vertikal fortgesetzt werden.

29. Zeichnen geometrischer Formen **4 Punkte**

Gegeben sei die wie folgt definierte Datenstruktur `Shape` geometrischer Formen:

```
type Radius = Float
type Side   = Float
type Vertex = (Float,Float)

data Shape =  Rectangle Side Side     | Ellipse Radius Radius
            | RtTriangle Side Side    | Polygon [Vertex]
            deriving Show
type ColoredShapes = [(Color,Shape)]
```

Schreiben Sie Funktionen

(a) `shapeToGraphic :: Shape -> Graphic`, die eine geometrische Form in eine Graphik überführt, die mit `drawInWindow` in einem Fenster angezeigt werden kann, / 2

(b) `drawShapes :: Window -> ColoredShapes -> IO ()`, die eine Reihe von gefärbten Graphiken in einem Fenster ausgibt. / 2

Verwenden Sie zum Umrechnen der Pixelkoordinaten eines Graphikfensters in Gleitkommakoordinaten eines Koordinatensystems, dessen Ursprung in der Mitte des Graphikfensters liegt, die im Modul `Trans.hs` (siehe Vorlesungsseite) zur Verfügung gestellten Konvertierungsfunktionen:

```
module Trans (
    trans,     -- :: Vertex -> Point
               -- Die Koordinaten eines Koordinatenpunktes werden
               -- in Pixel umgerechnet.
    transList, -- :: [Vertex] -> [Point]
               -- Umrechnung einer Liste von Koordinatenpunkten in
               -- Pixelpunkte.
    xWin, yWin -- :: Int
               -- Festlegung der Fenstergroesse in Pixeln
    ) where ...
```

Rechtecke und Ellipsen sollen so positioniert werden, dass ihr Mittelpunkt dem Ursprung des Koordinatensystems entspricht. Rechtwinklige Dreiecke sollen im rechten oberen Quadranten des Koordinatensystems erscheinen, wobei die Ecke mit dem rechten Winkel im Ursprung liegt.