Prof. Martin Hofmann, PhD
Dr. Steffen Jost

Ludwig-Maximilians-Universität München
Institut für Informatik
28. Mai 2015

6. Übung zur Vorlesung

Programmierung und Modellierung

A6-1 ***Compose*** Alois Dimpfelmoser möchte eine Funktion programmieren, welche die Summe der Quadrate aller geraden Zahlen aus einer Liste berechnet. Weil Alois der pointfree-Stil so gut gefällt (komischerweise sogar besser als List-Comprehension), hat er unter Verwendung von `compose` aus Folie 06-25 folgendes dazu implementiert:

```
geradequadratsumme  =  compose [sum, map (^2), filter even]
```

Leider mag GHC diese Definition nicht! Helfen Sie den armen Alois!

Wo liegen der/die Fehler? Wie lautet die richtige Definition im pointfree-Stil?

A6-2 ***I need a Dollar*** Ein klammer Kleptomane hat alle `$` geklaut! Fügen Sie in die nachfolgenden Haskell-Ausdrücke wieder `$` ein, so dass jeder Ausdruck zu `42` auswertet!

Hinweis: Es ist ausschließlich `$` einzufügen; sonst nichts, auch keine Klammern! Die Infix-Funktion `($)` wurde auf Folie 06-28 besprochen. Wem unklar ist, wie die Aufgabe anzugehen ist, kann die ersten beiden Teilaufgaben auch zuerst durch Einfügen von runden Klammern lösen, und diese dann erst anschließend wieder durch `$` ersetzen; bei den letzten beiden Teilaufgaben geht das aber nicht mehr — warum?

a) `div 169 3 + 1`

b) `sum filter even [3..11] ++ [13..15]`

c) `(2)(*21)`

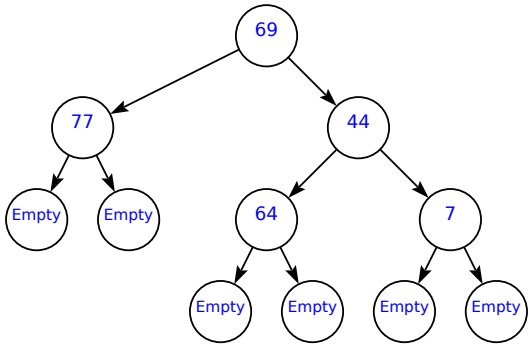
d) `(foldr) (6) (6) [(-),(*),(-),(+)]`

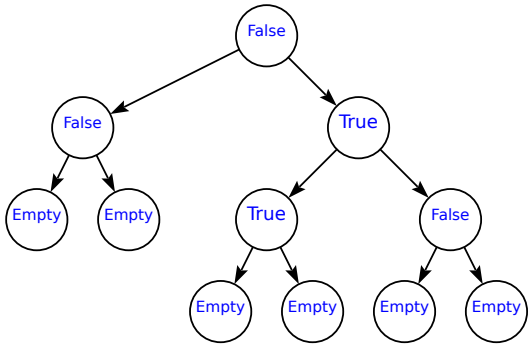
A6-3 ***Bäume*** Gegeben sind folgende Deklarationen: Record-Syntax: Folie 04-12

```
data Tree a = Empty | Node { label :: a, left,right :: Tree a }

leaf :: a -> Tree a
leaf a = Node a Empty Empty
```

a) Deklarieren Sie eine Konstante `myTree :: Tree Int`, welche den hier links abgebildeten Baum repräsentiert:





b) Schreiben Sie eine Funktion `myFmap :: (a -> b) -> Tree a -> Tree b` welche eine beliebige Funktion auf jede Knotenmarkierung eines Baumes anwendet. (Die Knotenmarkierung ist der Wert, welcher im `label`-Feld eines Baumknotens gespeichert wird.)

So wie `map :: (a -> b) -> [a] -> [b]` die Länge einer Liste unverändert lässt, so soll auch `myFmap` die Struktur des Baumes ebenso unverändert lassen.

Beispiel: `myFmap even myTree` sollte den rechts abgebildeten Baum zurückliefern.

c) Machen Sie `Tree` zu einer Instanz der Typklasse `Functor` aus der Standardbibliothek!

```
class Functor f where
    fmap :: (a -> b) -> f a -> f b
```

Hinweis: Diese Teilaufgabe ist sehr einfach, wenn Sie `myFmap` wiederverwenden und nicht weiter darüber nachdenken. *Zusatzfrage:* Wenn Sie aber darüber nachdenken, dann ist diese Typklasse etwas merkwürdig. Warum?

H6-1 ***Abstiegsfunktion IV*** (3 Punkte; Abgabeformat: Text oder PDF)
Beweisen Sie, dass folgende Funktion von Folie 04-16 terminiert. Sie dürfend dabei zur Vereinfachung annehmen, dass `(++)` immer terminiert. (`(:)` terminiert als Konstruktor sowieso.)

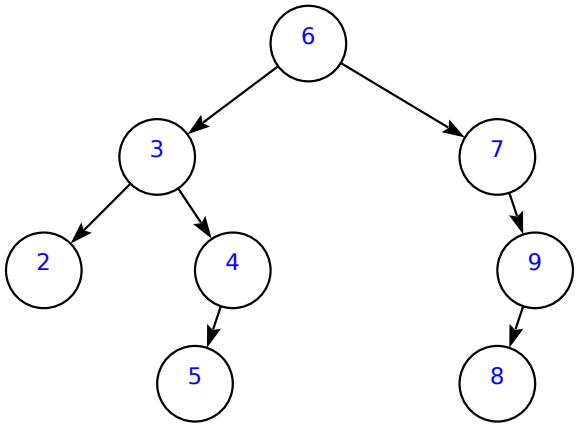
```
data Baum = Blatt Char | Knoten Baum Char Baum

dfCollect :: Baum -> String
dfCollect (Blatt c) = [c]
dfCollect (Knoten links c rechts) = c : dfCollect links ++ dfCollect rechts
```

H6-2 ***Bäume Drucken*** (4 Punkte; Datei `H6-2.hs` als Lösung abgeben)
Machen Sie den Typ `Tree a` aus Aufgabe A6-3 zu einer Instanz der Typklasse `Show`, unter der Voraussetzung, dass `a` bereits ebenfalls der Typklasse `Show` angehört. Einfacher ausgedrückt: Wandeln Sie Bäume in Ascii-Art um!

Beispiele: Zwei verschiedene Lösungsmöglichkeiten für den Baum in der Mitte:[1]

```
   /-9
  |   \-8
 /-7
6
|   /-4
|  |   \-5
 \-3
   \-2
```




```
6
|-7
| |-9
|   \-8
\-3
  |-4
  | \-5
  \-2
```

Auf der Vorlesungshomepage finden Sie eine Dateivorlage `H6-2.hs`. welche eine sehr ähnliche, mehrzeilige und eingerückte Ausgabe für die Listen aus A4-1 als Beispiel demonstriert.

[1] Wir verzichten auf die Ausgabe von `Empty`; Sie können dafür `*` ausgeben, wenn Sie dies einfacher finden.

- Unverpflichtende Ratschläge
  - Es ist vermutlich einfacher, den Baum auf der Seite liegen auszugeben.
  - Jede Knotenmarkierung wird in eine eigene Zeile geschrieben, so dass die Länge der Knotenmarkierung unproblematisch ist.
  - Je tiefer ein Knoten im Baum ist, desto weiter rechts wird er gedruckt. Die Funktion zum Drucken sollte also als zusätzliches Argument die aktuelle Tiefe mitführen.
- Bewertungsrelevant für volle Punktzahl
  - Eine Ausgabe ohne durchgezogene Linien (also Beispiele ohne `|`) bringt nicht die volle Punktzahl, ist aber für einen Anfang deutlich einfacher.
  - Es sollte klar erkennbar sein, ob es sich jeweils um einen linken oder rechten Teilbaum handelt, also `t2` und `t3` aus der Vorlage sollten klar unterscheidbar sein.

H6-3 ***Bäume Zeichen*** (2 Punkte; Datei `H6-3.hs` als Lösung abgeben)
Stellen Sie beliebige binäre Bäume als Vektorgrafik dar!

Auf der Vorlesungshomepage finden Sie eine Dateivorlage `H6-3.hs`, welche als Beispiel zeigt, wie Sie eine Vektorgrafik eines Baumes mit 3 Knoten erzeugen. Die Bibliothek `diagrams` nimmt uns einen Großteil der Arbeit ab. Leider ist diese nicht in Standardbibliothek enthalten. Geben Sie zur Installation in eine Konsole ein (die Installation kann eine ganze Weile dauern):

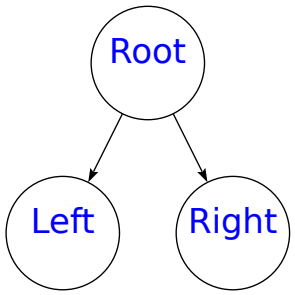
```
> cabal update
Downloading the latest package list from hackage.haskell.org
> cabal install diagrams
```

Das Tool `cabal-install` sollte bereits mit der Haskell-Plattform auf Ihren Rechner installiert worden sein. Falls Sie Probleme mit der lokalen Installation haben, dann verwenden Sie einfach die Rechner im CIP-Pool. Dies ist auch aus der Ferne möglich.[2]

Zur Lösung dieser Aufgabe müssen Sie *nicht* die Dokumentation von `diagrams` lesen; die Vorlage enthält bereits alles, was Sie zur Lösung dieser Aufgabe benötigen:

```
diagNode :: String -> Diag
diagNode s = text s `atop` circle 1

diagTriangle :: Diag
diagTriangle = connectOutside "X" "L" $
               connectOutside "X" "R" $
          nx
          ===
      (nl ||| nr) # center
  where
    nx = diagNode "Root"  # named "X"
    nl = diagNode "Left"  # named "L"
    nr = diagNode "Right" # named "R"
```



[2] Informationen zum Remote-Login am CIP-Pool finden Sie auf: http://www.rz.ifi.lmu.de/FAQ/index.html im Abschnitt "Von zu Hause/remote aus..."

Die besondere Einrückung hier ist ohne Bedeutung. Werte des Typs `Diag` sind Diagramme, welche wir miteinander zu größeren Diagrammen kombinieren können:

- `diagNode s` zeichnet einen Text `s` innerhalb eines Kreises.
- `x `atop` y` kombiniert Diagramme, zeichnet `x` über `y` drüber.
- `x === y` kombiniert zwei Diagramme, wobei `x` oberhalb von `y` platziert wird.
- `x ||| y` kombiniert zwei Diagramme, wobei `y` rechts von `x` platziert wird.
- `named :: String -> Diag -> Diag` gibt einen Diagramm einen Namen
- `connectOutside :: String -> String -> Diag -> Diag` zeichnet einen Pfeil zwischen benannten Teildiagrammen. Diese Teildiagramme müssen in dem übergeben Diagramm bereits mit den angegebenen Namen enthalten sein.
- `(#) :: a -> (a -> b) -> b` füttert ein Argument an eine gegebene Funktion. Dabei ist `x # f` identisch zu `f $ x`. Diese Infix-Funktion `(#)` aus `diagrams` dient wie `($)` auch lediglich zur hübschen Formatierung unseres Codes und spart Klammern.

Die Vektorgrafik wird dann durch Ausführen des Codes erzeugt, wobei die `main` Funktion der Vorlage nicht mehr verändert werden muss:

```
> ghc H6-3.hs
[1 of 1] Compiling Main             ( H6-3.hs, H6-3.o )
Linking H6-3 ...
> ./H6-3 -o Demo.svg -h 640 -S Triangle
> firefox Demo.svg
```

Kompilieren des Codes mit `ghc` erzeugt eine ausführbare Datei. Diese führen wir dann einmal aus,[3] wobei wir als Parameter den Namen der Ausgabedatei, die Höhe der Grafik, und die Auswahl des zu berechnenden Diagramms angeben (ansatt `-S Triangle` später also z.B. `-S Tree1` eintippen). Danach können wir uns die Grafik in einem Webbrowser anschauen.[4] Es ist Ihnen überlassen, ob Sie leere Blattknoten anzeigen möchten wie in den Bildern zu A6-3, oder nicht, wie etwa in der Grafik zu H6-2.

Hinweis: Die Bibliothek `diagrams` ist nicht prüfungsrelevant. Sie ist aber auch gar nicht der Inhalte dieser Aufgabe! Zur Lösung dieser Aufgabe müssen wir hier lediglich mit Bäumen und Funktionen umgehen — was natürlich prüfungsrelevant ist.

Abgabe: Lösungen zu den Hausaufgaben können bis **Dienstag, den 09.06.2015, 11:00 Uhr** mit UniworX abgegeben werden.

Aufgrund des Klausurbonus müssen die Hausaufgaben von Ihnen alleine gelöst werden. Abschreiben bei den Hausaufgaben gilt als Betrug und führt zum Klausur-Ausschluss.

[3] Je nach OS alternativ auch in einem Schritt mit: `runghc Diagramm.hs -o Demo.svg -h 640 -S Triangle`

[4] Mac-User geben zur Ansicht der Datei ein: `open -a firefox Demo.svg`