Technische Universität München
Institut für Informatik
Prof. Tobias Nipkow, Ph.D.
Lars Noschinski, Lars Hupel, Dr. Jasmin Blanchette

WS 2013/14
28.01.2013
Abgabe: 04.02.2013, 15:30

Einführung in die Informatik 2
14. Übung

Aufgabe G14.1 Endrekursive Funktionen I

Entscheiden Sie für die folgenden Funktionen, ob diese endrekursiv sind:

1.
```
prod :: Num a => a -> [a] -> a
prod n [] = n
prod n (m:ms) = prod (n*m) ms
```

2.
```
prod :: Num a => [a] -> a
prod [] = 1
prod (m:ms) = if m = 0 then 0 else m * prod ms
```

3.
```
prod :: Num a => a -> [a] -> a
prod n [] = n
prod n (m:ms) = if m == 0 then 0 else prod (n*m) ms
```

Aufgabe G14.2 Endrekursive Funktionen II

Wir betrachten die Funktion `concat :: [[a]] -> [a]`, die eine Liste von Listen nimmt und diese konkateniert:

```
concat [[1,2],[],[5,6],[7]] = [1,2,5,6,7]
```

Geben Sie eine endrekursive Implementierung von `concat` an.

Aufgabe G14.3 Reduktion

Werten Sie die folgenden Ausdrücke mit Haskells Auswertungsstrategie vollständig aus:

```
map (*2) (1 : threes) !! 1
(\f -> \x -> x + f 2) (\y -> y * 2) (3 + 1)
head (filter (/=3) threes)
```

Welche Auswertungen terminieren nicht? Dabei seien die verwendeten Funktionen wie folgt definiert:

```
map _ [] = []
map f (x:xs) = f x : map f xs

filter _ [] = []
```

```
filter f (x:xs) | f x = x : filter f xs
                | otherwise = filter f xs

(x:xs) !! n | n == 0 = x
            | otherwise = xs !! (n - 1)

threes = 3 : threes
```

Aufgabe H14.1 QuickCheck (10 Punkte)

Gegeben sei eine Ihnen unbekannte Implementierung der Funktion

```
uniqueElems :: Eq a => [a] -> [a]
```

Angeblich liefert `uniqueElems xs` eine duplikatfreie Liste der Elementen aus der Liste `xs`, in einer beliebigen Reihenfolge. Überprüfen Sie diese Behauptung, indem Sie eine vollständige Sammlung von QuickCheck-Tests für `uniqueElems` definieren. Eine Sammlung von Tests ist *vollständig*, wenn im Falle eines Fehlers in der Implementierung es Eingaben für die Tests gibt, die den Fehler aufdecken, d.h. der Test schlägt fehl. (Ob QuickCheck tatsächlich innerhalb von n Iterationen diese Eingabe generiert, ist eine andere Frage.)

Hinweis: Es werden diesmal keine automatischen Tests zur Verfügung gestellt. Testen Sie Ihre QuickCheck-Tests gegen korrekte und falsche Implementierungen von `uniqueElems`.

Aufgabe H14.2 Brisante Benennung Boolescher Bedingungen (10 Punkte)

Gegeben sei ein Datentyp zur Darstellung von einfachen booleschen Formeln:

```
data Fml a = Var a | Neg (Fml a) | Conj [Fml a] | Disj [Fml a]
  deriving (Eq, Show)
```

Eine Formel ist also entweder eine boolesche Variable, die Negation einer Formel, die Konjunktion von null oder mehr Formeln oder die Disjunktion von null oder mehr Formeln.

Implementieren Sie eine Funktion `rename :: (a -> a) -> Fml a -> Fml a`, die die Variablen einer Formel durch die Anwendung des ersten Arguments (der *Umbenennungsfunktion*) systematisch umbenennt.

Beispiel: Die boolesche Formel $\neg x \vee y$ wird als `Conj [Neg (Var 'x'), Var 'y']` dargestellt. Für die Umbenennungsfunktion `Data.Char.toUpper` soll `rename` die Formel $\neg X \vee Y$ zurückgeben:

```
rename Data.Char.toUpper (Conj [Neg (Var 'x'), Var 'y']) ==
    Conj [Neg (Var 'X'), Var 'Y']
```

Aufgabe W14.1 Für die van Goghs dieser Welt (**0 Punkte**, Wettbewerbsaufgabe)

In dieser allerletzten Wettbewerbsaufgabe geht es darum, ein Pixel- oder Vektor-Bild in einem Standardformat (z.B. PNG, SVG, PDF) zu generieren. Alle Graphikbibliotheken von Drittanbietern sind ausdrücklich erlaubt. Siehe zum Beispiel:

`http://www.cs.yale.edu/homes/hudak/SOE/`
`http://www.haskell.org/haskellwiki/Applications_and_libraries/Graphics`

Die Bewertungskriterien sind Ästhetik (*hübsches Bild!*) und Technik (*toller Code!*). Die Jury besteht aus dem Meta-Master und den (Co(ntra)-)Masters of Competition.

Diese Wettbewerbsaufgabe ist getrennt in einer Datei namens `Gogh.hs` einzureichen. Zusammen mit `Gogh.hs` sollten sie auch das generierte Bild einreichen unter den Namen `gogh.`*`xxx`*, wo *`xxx`* z.B. `png` ist.

Wichtig: Wenn Sie diese Aufgabe als Wettbewerbsaufgabe abgeben, stimmen Sie zu, dass Ihr Name ggf. auf der Ergebnisliste auf unserer Internetseite veröffentlicht wird. Sie können diese Einwilligung jederzeit widerrufen, indem Sie eine Email an `fp@fp.in.tum.de` schicken.