SoSe 2014

Prof. Dr. Margarita Esponda

ProInformatik II: Funktionale Programmierung

3. Übungsblatt fürs Wochenende

Abgabe: Montag um 8:55

Abgabe: Montag pünktlich um 8:55 Uhr vor Vorlesungsbeginn in **Papierform**.

1. Aufgabe (3 Punkte)

In dem Haskell-Prelude ist die iterate-Funktion wie folgt definiert:

```
iterate :: (a -> a) -> a -> [a]
iterate f x = x : iterate f ( f x )
```

Definieren Sie unter Verwendung der iterate-Funktion und anonymer Funktionen (Lambda-Ausdrücken) Funktionen, die folgende Listen erzeugen können:

[True, False, True, False, True, ...]

[2, -4, 8, -16, 32, -64, 128, -256, 512, ...]

[2, 6, 12, 20, 30, 42, 56, 72, 90, 110, ...]

2. Aufgabe (6 Punkte)

Eine **unfold**-Funktion, die ein einfaches rekursives Pattern, um eine Liste zu produzieren, implizit darstellt, kann wie folgt definiert werden.

```
unfold p f g x      | p x = []
                    | otherwise = f x : unfold p f g (g x)
```

Redefinieren Sie unter Verwendung der **unfold**-Funktion folgende Funktionen

(map f), (iterate f) und int2bin

3. Aufgabe (4 Punkte)

Verwandeln Sie folgenden Lambda-Ausdruck in einen SKI-Ausdruck. Alle Zwischenschritte sollen aufgeschrieben werden so wie die Regel, die dafür verwendet worden sind.

λx.y(yxy)

4. Aufgabe (6 Punkte)

Wir haben in der Vorlesung mögliche Darstellungen von Listen als Lambda-Abstraktionen für folgende Funktionen definiert:

head, tail, (:), NIL, L (leere Liste)

Schreiben Sie damit die reverse- und (++)-Funktion für Listen.

5. Aufgabe (5 Punkte)

Erweitern Sie den SKI-Parser aus der Vorlesung, sodass analog zum Lambda-Parser auch ***longmacros*** übersetzt werden können. Testen Sie Ihren Parser mit mindestens 3 vordefinierten Funktionen.

Anwendungsbeispiele:

```
parser  "x"          =>   Var "x"
parser  "{1}KK"    =>   App (App (App (App S (App (App S (App K S)) (App (App S
                                (App K K)) I))) (App K I)) K) K
```

6. Aufgabe (8 Punkte)

Definieren Sie analog zu der ersten eval-Funktion für Lambda-Ausdrücke aus der Vorlesung eine eigene **eval**-Funktion für SKI-Ausdrücke. Testen Sie Ihre **eval**-Funktion mit folgenden Ausdrücken:

Anwendungsbeispiele:

skii "(S(SI(K(KI)))(K(S(KK)I)))(S(KK)I)" => "KI"

skii "(S(S(KS)(S(S(KS)(S(KK)I))(KI)))(K(K(KI))))(S(KK)I)(KI)" => "KI"

mit **skii** = **eval . parser**

(siehe vorgegebene Funktionen in Ressources -> Material -> SKII_U10_start_functions.hs.zip)

7. Aufgabe (3 Punkte)

Zeigen Sie, dass für alle $n \geq 1$ folgende Gleichung gilt:

$$\sum_{i=1}^{n} i \cdot i! = (n+1)! - 1$$

8. Aufgabe (4 Punkte)

Betrachten Sie folgende Definition der **powset**- und **length**-Funktion

```
powset          ::  [a] -> [[a]]
powset []       =  [[]]
powset (x:xs)  =  powset' ++ [x:ys | ys <- powset']
                     where
                          powset' = powset xs

length          :: [a] -> Integer
length  []       =  0
length  (x:xs)  =  1 + length xs

(++)            :: [a] -> [a] -> [a]
(++)   []      ys       = ys
(++)  (x:xs)  ys       = x : (xs ++ ys)
```

Beweisen Sie damit folgende Gleichung:

$$length\ (powset\ xs) = 2^{(length\ xs)}$$

Sie dürfen voraussetzen, dass folgende Hilfseigenschaft gilt:

length xs = length [z | x <- xs] mit z gleich einem beliebigen Ausdruck

9. Aufgabe (7 Punkte)

Zeigen Sie mittels vollständiger Induktion über **n**, dass folgende zwei Funktionsdefinitionen äquivalent sind.

```
maxSurfs  0  = 1
maxSurfs  n  = maxSurfs (n - 1) + n

maxSurfs' n  = aux 0 n
                  where
                    aux  acc  0 = acc + 1
                    aux  acc  n = aux (acc + n) (n-1)
```

10. Aufgabe

Fangen Sie an sich für die Klausur vorzubereiten.

Wichtige Hinweise:

1) Verwenden Sie geeignete Namen für Ihre Variablen und Funktionsnamen, die den semantischen Inhalt der Variablen oder die Semantik der Funktionen wiedergeben.

2) Verwenden Sie vorgegebene Funktionsnamen, falls diese angegeben werden.

3) Kommentieren Sie Ihr Programme.

4) Verwenden Sie geeignete lokale Funktionen und Hilfsfunktionen in Ihren Funktionsdefinitionen.

5) Schreiben Sie in allen Funktionen die entsprechende Signatur.