WS 2017/2018

Funktionale Programmierung

8. Übungsblatt

Prof. Dr. Margarita Esponda

(Abgabetermin: Mi., den 20.12., um 10:10, Hörsaal)

1. Aufgabe (4 Punkte)

Betrachten Sie folgende Funktionsdefinitionen:

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

(++)  []  ys  = ys
(++) (x:xs) ys = x:(xs ++ ys)
```

Beweisen Sie mittels struktureller Induktion über **xs** folgende Eigenschaft:

```
map f xs  ++  map f ys  ==  map f (xs ++ ys)
```

2. Aufgabe (12 Punkte)

Betrachten Sie folgende Funktionsdefinitionen:

```
(++)   []       ys         = ys
(++)  (x:xs)  ys         = x : (xs ++ ys)

reverse   []               =  []
reverse  (x:xs)          =  reverse xs  ++  [x]

elem  x  []                          =  False
elem  x  (y:ys) | x==y          =  True
                      | otherwise     =  elem  ys

dropWhile p []  =  []
dropWhile p (x:xs)  =  if  p x
                         then dropWhile p xs
                         else (x:xs)

takeWhile p []  =  []
takeWhile p (x:xs)   = if p x
                        then x:(takeWhile p xs)
                        else []
```

Beweisen Sie mittels struktureller Induktion über die Liste **xs**, dass für jede endliche Listen **xs** und **ys** folgende Gleichungen gelten:

a) (takeWhile p xs) ++ (dropWhile p xs) = xs

b) reverse (xs ++ ys) = reverse ys ++ reverse xs

c) elem a (xs ++ ys) = elem a xs || elem a ys

3. Aufgabe (6 Punkte)

Betrachten Sie folgende Funktionsdefinitionen:

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

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

Zeigen Sie mittels vollständiger Induktion über **n**, dass die Funktionen **maxPieces** und **maxPieces'** äquivalent sind.

4. Aufgabe (4 Punkte)

Betrachten Sie folgende Definition der **powerset** Funktion

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

Beweisen die Gültigkeit 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 e.1

5. Aufgabe (6 Bonuspunkte)

Unter Verwendung folgender Funktionsdefinitionen

```
data Tree a = Leaf a | Node  (Tree a)  (Tree a)
sumLeaves  ::  Tree a   ->  Integer
sumLeaves   (Leaf x)   = 1
sumLeaves   (Node  lt  rt)     =  sumLeaves  lt  +  sumLeaves  rt

sumNodes :: Tree a -> Integer
sumNodes   (Leaf x)          =  0
sumNodes   (Node  lt rt)     =  1 +  sumNodes  lt   +   sumNodes  rt
```

Beweisen Sie, dass für alle endlichen Bäume t :: Tree a   gilt:

sumLeaves t = sumNodes t + 1

6. Aufgabe (8 Bonuspunkte)

Betrachten Sie folgende Variante des algebraischen Datentyps der 3. Aufgabe und folgende Funktionsdefinitionen:

```
data Tree a = Nil | Node a (Tree a) (Tree a) | Leaf a

sumTree :: (Num a) => Tree a -> a
sumTree Nil              = 0
sumTree (Leaf x)         = x
sumTree (Node x l r)     = x + sumTree l  +  sumTree r

tree2list :: (Num a) => Tree a -> [a]
tree2list Nil            = []
tree2list (Leaf x)       = [x]
tree2list (Node x l r)   = tree2list l ++ [x] ++ tree2list r

sum :: (Num a) => [a] -> a
sum []       = 0
sum (x:xs)   = x + sum xs
```

Beweisen Sie, dass folgende Eigenschaft gilt:

sum.tree2list = sumTree

Wichtige Hinweise:

1. Schreiben Sie in alle Funktionen die entsprechende Signatur.
2. Verwenden Sie geeignete Namen für Ihre Variablen und Funktionsnamen, die den semantischen Inhalt der Variablen oder die Semantik der Funktionen wiedergeben.
3. Verwenden Sie vorgegebene Funktionsnamen, falls diese angegeben werden.
4. Kommentieren Sie Ihre Programme.
5. Verwenden Sie geeignete lokale Funktionen und Hilfsfunktionen in Ihren Funktionsdefinitionen.