WiSe 2015/2016

Funktionale Programmierung

12. Übungsblatt (Abgabe: Mi., den 27.Januar, um 10:10 Uhr)

Prof. Dr. Margarita Esponda

Ziel: Beweisen von Programmeigenschaften.

1. 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.

2. 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

3. Aufgabe (6 Punkte)

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

4. Aufgabe (8 Punkte)

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