WS 2020/2021

Funktionale Programmierung

12. Übungsblatt

Prof. Dr. Margarita Esponda

1. Aufgabe (8 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
```

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

a) reverse (reverse xs) = xs

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

2. 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.

3. 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

4. Aufgabe (4 Punkte)

Betrachten Sie folgende Funktionsdefinition:

```
f = flip g
```

Beweisen Sie mittels struktureller Induktion über **xs**, dass

```
foldl g z xs = foldr f z (reverse xs)
```

5. Aufgabe (4 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
```

6. Aufgabe (6 Punkte)

Betrachten Sie folgende algebraische Datentyps und 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
```

Wichtiger Hinweis:

Die Lösungen sollen elektronisch (nur Whiteboard-Upload) abgegeben werden. Es ist k**eine verspätete Abgabe per Email erlaubt.**