**FP-Aufgaben 06**

Aufgabe 1

Welchen Typ haben die folgenden Funktionen?

* **foldr (-)**
* **(+2)**
* **f [] = f [1]**
* **f x = (\x -> x) x**
* foldr (-) :: Num b => b -> [b] -> b
* (+2) :: Num a => a -> a
* f :: Num a => [a] -> [a]

f [ ] = f [1]

* f :: generic -> generic

f x = (\x -> x) x

Aufgabe 2

Verwenden Sie die Funktion foldr, um die Funktion map' zu definieren:

map' :: (a -> b) -> [a] -> [b]

map' \_ [ ] = [ ]

map' f (x:xs) = f x : map' f xs

foldrMap :: (t -> a) -> [t] -> [a]

foldrMap \_ [] = []

foldrMap f xs = foldr (\el els -> f el : els) [] xs

Aufgabe 3

a)

Implementieren Sie eine rekursive Funktion take\_while, die ein Prädikat und eine Liste als Parameter hat und das längste Anfangsstück der Liste zurückgibt, dessen Elemente alle das Prädikat erfüllen.

Zum Beispiel soll der Aufruf take\_While (\x -> x < 3) [1,2,1,1,4,2,5] zu [1,2,1,1] ausgewertet werden

take\_while :: (a -> Bool) -> [a] -> [a]

take\_while \_ [] = []

take\_while pred (x : xs) =

if (pred x)

then x : take\_while pred xs

else []

b)

Schreiben Sie das Programm mittels foldr um.

take\_while' :: (a -> Bool) -> [a] -> [a]

take\_while' pred xs =

foldr

( \el els ->

if pred el

then el : els

else []

)

[]

xs

Aufgabe 4

Schreiben Sie eine Funktion, die alle Leerzeichen aus einem Text löscht und den ersten Buchstaben jedes Wortes in einen Großbuchstaben umwandelt.

z.B. wäre der Text „Das ist ein Text“, soll die Ausgabe der Funktion „DasIstEinText“ sein.

removeSpace' :: [Char] -> [Char]

removeSpace' [] = []

removeSpace' (x : xs) =

foldr

( \el els ->

if el /= ' '

then el : els

else toUpper (head els) : tail els

)

[]

(toUpper x : xs)