Inhaltsverzeichnis

1 Listenfunktionen

1.1 ++

Konkatiniert 2 Liste.
Bsp.:
[1,2] ++ [3,4]
[1,2,3,4]

1.2 head

Gibt das Kopfelement zurück.
Bsp.:
head [1,2,3]
1

1.3 tail

Gibt die Liste ohne das Kopfelement zurück.
Bsp.:
tail[1,2,3]
[2,3]

1.4 take

Gibt die ersten N Elemente zurück.
Bsp.:
take 2 [1,2,3]
[1,2]

1.5 drop

Gibt die Liste ohne die ersten N Elemente zurück.
Bsp.:
drop 2 [1,2,3]
[3]

1.6 length

Gibt die Länge der Liste zurück.
Bsp.:
length [1,2,3]
3

1.7 sum

Gibt die Summe aller Einträge der Liste zurück.
Bsp.:
sum [1,2,3]
6

1.8 product

Gibt das Produkt aller Einträge der Liste zurück.
Bsp.:
product [1,2,3]
6

1.9 inverse

Gibt umgekehrte Liste zurück.
Bsp.:
inverse [1,2,3]
[3,2,1]

1.10 zip

Gibt eine Liste bestehend aus Tupeln der sich entsprechenden Elemente der 2 Eingangslisten.
Bsp.:
zip [1,2,3] [4,5,6]
[(1,4),(2,5),(3,6)]

1.11 zipWith

Bildet eine Liste der Funktionsanwendung auf die sich entsprechenden Elemente der 2 Eingangslisten.
Bsp.:
zipWith (+) [1,2,3] [4,5,6]
[5,7,9]

1.12 map

Bildet eine Liste von Werten auf eine Liste ihrer Funktionswerte ab.
Bsp.:
map (1+) [2,3,4]
[3,4,5]

1.13 filter

Gibt eine Liste bestehend aus den Werten der Eingangsliste, die die Filterbedingung erfüllen, zurück
Bsp.:
filter (<3) [1,2,3]
[1,2]

1.14 takeWhile

Nimmt alle Elemente einer Eingangsliste bis zu der Stelle, an der die Bedingung das erste mal nicht erfüllt ist.
Bsp.:
takeWhile (<3) [1,2,3,2,1]
[1,2]

1.15 dropWhile

Nimmt alle Elemente einer Eingangsliste ab der Stelle, an der die Bedingung das erste mal nicht erfüllt ist.
Bsp.:
dropWhile (<3) [1,2,3,2,1]
[3,2,1]

1.16 foldr

Bibliotheksfunktion zur Vereinfachung von Funktionen der Form:
fkt [] = c
fkt (x:xs) = x ‘op‘ fkt xs
Bsp.:
sum = foldr (+) 0
sum’ [] = 0
sum’ (x:xs) = x + sum’ xs