Altklausur Bry 2012 Remastered für Programmierung und Modellierung 2016

Alexander Isenko

July 17, 2016

Besprechung am 22.Juli 2016

Aufgabe 1

Hier wird nach Funktionen gefragt die ihr euch selber einfallen lassen könnt, total egal wie kompliziert oder einfach sie sind, sie müssen lediglich den Bedingungen entsprechen.

a) Definieren sie eine monomorphe Funktion (keine Typvariablen, nur feste Typen)

b) Definieren sie eine polymorphe Funktion (mit Typvariablen)

   i) parametrisch polymorph (keine Typvariablen vor (=>))

   ii) ad-hoc polymorph (mid. eine Typvariable vor (=>))

c) Definieren sie eine *gecurriete* Funktion (eine Funktion die ein Tupel nimmt, aber mit `curry` stattdessen die Argumente hintereinander akzeptiert)

```
1  -- Hilfestellung:
2
3      > :t curry
4      curry :: ((a, b) -> c) -> a -> b -> c
```

Aufgabe 3

Definieren sie die Funktion `reverse` für Listen in Haskell.

```
> :t reverse
[a] -> [a]

> reverse "hallo"
"ollah"

> reverse [1,2,3]
[3,2,1]

> reverse []
[]
```

Aufgabe 4

Sei folgender Code gegeben, was ist das Ergebnis von `res`?

```
y = 5
x = 2
goo y     = x * y
fuu (x,y) = x + goo y
res = (goo y, fuu (5,7))
```

Aufgabe 6

Definieren sie das Standartskalarprodukt mithilfe von `zip` und `Data.List.foldl/foldr`

```
--  Typsignatur:
--
skalarProdukt :: Num a => [a] -> [a] -> a
--
--  Beispiel:
--
skalarProdukt [1,2,3] [4,5,6] = 1*4 + 2*5 + 3*6 = 4 + 10 + 18 = 32

-- Hilfestellung:
--
zip :: [a] -> [b] -> [(a,b)]
zip []     []     = []
zip (x:xs) (y:ys) = (x,y) : zip xs ys


```

```haskell

foldl :: (b -> a -> b) -> b -> [a] -> b
foldl f acc []     = acc
foldl f acc (x:xs) = foldl f (f acc x) xs

foldr :: (a -> b -> b) -> b -> [a] -> b
foldr f acc []     = acc
foldr f acc (x:xs) = f x (foldr f acc xs)
```

Aufgabe 7

Sei folgender Code gegeben.

```haskell
ks = [3,5,7]

pred = \x -> x < 0

f p x (y:ys) = if p y
                  then f (not . p) x ys
                  else f p         x ys
f p x _      = p x

g x = let f x = x
      in g (f x)
```

Geben sie wenn möglich den Wert den Typ der folgenden Ausdrücke an:

a) `f`

b) `f pred 0 []`

c) `f pred 0 ks`

d) `g`

e) `g ks`