Algorithmen und Programmierung I

WS06/07
E. Fehr

Musterlösung

Aufgabe 1a)

```
type Artikel = String  --  ein Artikel wird durch seinen Namen angegeben
type Preis = Int --  Preise werden in Cent ausgewiesen
type Eintrag = (Artikel, Preis) --  ein Eintrag im Sortiment
waren :: Sortiment  --  das aktuelle Sortiment des Geschäftes
waren = [(``Puppe'',1999), (``Stift'',25), (``Skat'',499)] --  Beispiel zum Testen
```

Aufgabe 1b)

```
preis :: Artikel  -> Preis --  liefert aktuellen Preis in Bezug auf akt. Sortiment
preis = preisSort waren
preisSort :: Sortiment  -> Artikel ->  Preis --  Preis eines Artikels bzgl. Sortiment

preisSort ((b,p):es) a
   | a==b      = p
   | otherwise = preisSort es a

preisSort [] a = error "Artikel nicht im Sortiment vorhanden."
```

Aufgabe 1c)

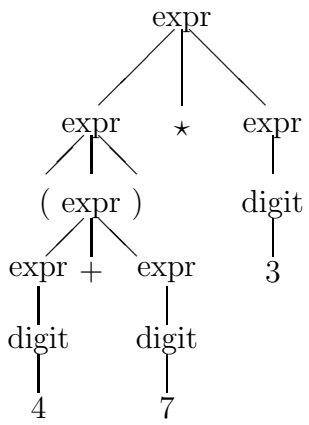
```
artikelAnzahl :: Int  --  Anzahl der Artikel im aktuellen Sortiment
artikelAnzahl = length waren
```

Aufgabe 1d)

```
teuersterArtikel :: Artikel  --  teuerster Artikel aus aktuellem Sortiment
teuersterArtikel = fst (maxEintrag waren)
maxEintrag :: Sortiment -> Eintrag  --  maximaler Eintrag bzgl. eines Sortiments
maxEintrag [] = error "Sortiment ist leer, daher gibt es keinen teuersten Artikel."
maxEintrag (e:es) = max' e es -- max' führt einen vorläufig maximalen Parameter mit
 max' :: Eintrag -> Sortiment ->  Eintrag  --  Akkumulatortechnik zur Bestimmung
                                               des Maximums

max' e [] = e
max' (a1,p1) ((a2,p2):es)
   | p1 >= p2  = max' (a1,p1) es
   | otherwise = max' (a2,p2) es
```

Aufgabe 2)



Aufgabe 3) $(\lambda x.xy)\lambda z.\ z \rightarrow (\lambda z.z)y \rightarrow y$

Zweimal β-Reduktion angewendet.

Aufgabe 4a)

```
sum1, sum2, sum3 :: Num a => [a] -> a

sum1 [] = 0
sum1 (x:xs) = x + sum1 xs

sum2 = sumAkk 0
   where sumAkk :: Num a => a -> [a] -> a
         sumAkk akk [] = akk
         sumAkk akk (x:xs) = sumAkk (akk+x) xs

sum3 = foldr (+) 0
```

Aufgabe 4b) Zu zeigen: `sum1` ℓ = `sum2` ℓ für alle ℓ.

Ind.Anfang: ℓ = `[]`

`sum1 []` = 0 `(sum1.1)`

`sum3 []` = `foldr (+) 0 []`

= 0 `(foldr.1)`

Ind.Schritt:

`sum1 (x:xs) = x + sum1 xs (sum1.2)`

`= x + sum3 xs (J.V.)`

`sum3 (x:xs) = foldr (+) 0 (x:xs) (sum3)`

`= (+) x (foldr (+) 0 xs) (foldr.2)`

`x + sum3 xs (sum3.2 rückwärts)`

Aufgabe 5a)

```
data Farbe = Rot | Gelb | Blau
   deriving (Eq, Ord, Show, Read)

data Mischfarbe = Grund Farbe | Misch Mischfarbe Mischfarbe
   deriving (Eq, Ord)


instance Show Mischfarbe where
   show (Grund f)     = [head (show f)]
   show (Misch f1 f2) = (show f1) ++ (show f2)
```

Aufgabe 6)

```
mapAndFilter :: (a->b) -> (b->Bool) -> [a] -> [a]
mapAndFilter f t xs = [ x | x <- xs, t(f x) ]
```