Programmierung und Modellierung Klausur SS17 - Gedankenprotokoll

Aufgabe 1

Geben sie den Typen der Ausdrücke an.

a) f x = x ++ „a“ ++ x

   [Char] -> [Char]

   Char -> Char

   ?????????

   Keins Davon: ……..

   Kein korrekter Haskell Ausdruck

b) (\x -> \y -> y:x)

   a -> [a] -> [a]

   a -> [b] -> [b]

   [a] -> a -> [a]

   Keins Davon: ……..

   Kein korrekter Haskell Ausdruck

c) x = „Ha“ ++ „N“ ++ „No“

   [Char] -> [Char]

   [a] -> [a]

   ???????

   Keins Davon: ……..

   Kein korrekter Haskell Ausdruck

d) h = “b” “k”

   [Char] -> [Char]

   ???????

   ???????

   Keins Davon: ……..

   Kein korrekter Haskell Ausdruck

Aufgabe 2

Gegeben sei die Funktion f.

f <= 0 = 0

f (1) = 1

f (2) = 1

f (x) = f(x-1) + f(x-2) + f(x-3)

a)

Definiere die rekursive Funktion welche in Aufgabe 2 beschrieben wird (nicht endrekursiv).

b)

Definiere die gleiche Funktion aus a) endrekursiv

Aufgabe 3

f = (\n -> if n <= 0 then n + 5 else doppelt (n-1))

eins = (\y -> 1)

doppelt = (\x -> x + x)

hd = (\(x:xs) -> x)

tl = (\(x:xs) -> xs)

a) Werte Applikativ aus.

   f 1

b) Werte Normal aus.

   f (eins (2))

c) Werte Verzögert aus

   doppelt (eins (f (5)))

d) Werte verzögert aus.

   hd (tl [4,5,6])

Aufgabe 4

data BB a = L | K (BB a) a (BB a)

a) Geben sie den ausgeglichenen Baum an, welcher die Elemente 0, 1, 2, 3, 4, 5, 6 enthält

b) Implementieren sie die Suchfunktion für diesen Baum

   suche :: a -> BB a -> Bool

   suche …….. …… = False

   suche …… …… = ……..

Aufgabe 5

a) Vervollständigen sie die tief-Funktion.

   tief :: a –> (b –> a –> a –> a) –> BB b –> a

   tief fK fL ……. = ……

   tief fK fL (K left w right) = fk (tief ….. fK ……..) w (tief fL …… fk ……)

b) Welche Funktion gibt an wie viele Blätter der Baum hat.

   tief 1 (\left w right -> left + right + 1) tree

   tief 0 (\left w right -> left + right + 1) tree

   tief 1 (\left w right -> left + right) tree

   tief 0 (\left w right -> 1 + left + right) tree

Aufgabe 6

a) Geben sie einen rekursiven Datentyp an um beliebig viele Email Adressen abzuspeichern. Und geben sie die Darstellung der Email Adressen, „a1“, „a2“ und „a3“ an.

b) Ergänzen sie die Funktion.

emailString :: Adressen -> String

emailString ….. =

emailString ….. =

c) Definieren sie den Datentype Kunde, (hat einen Namen, und beliebig viele Email-Adressen)

d) Implementieren sie Kaeufer für Kunde.

class Kaeufer a where

printKaeufer :: a -> String

instance Kaeufer Kunde where

……

……

……

Aufgabe 7 Monoid

data MB = W | F

a) Ergänzen sie die Funktion (logisches und)

und :: MB -> MB -> MB

und …… ……. = W

und …… ……. = F

und …… ……. = F

und …… ……. = F

b) Zeigen sie, dass bei der Funktion „und“ ein Neutrum existiert.

c) Zeigen sie, dass „und“ assoziativ ist.

d) Vervollständigen sie den Code, und zeigen sie wieso MB ein Monoid ist.

```
instance Monoid MB
        mempty = ….
        mappend = ….
```

Aufgabe 8

```
anwenden :: (a -> b -> c) -> t a -> t b -> t c
anwenden op sz1 sz2 = do
        z1 <- sz1
        z2 <- sz2
        return (op z1 z2)
```

a) Vervollständigen Sie die Funktion ioConcat.

```
IO String -> IO String -> (( IO String ))
ioConcat = (( anwenden (++) ))
```

b) Kreuzen sie an welche Aussagen zutreffen.

anwenden getLine getLine *liest zwei Zeilen ein*

anwenden getLine „ab“ *ist nicht korrekt*

anwenden getLine „abc“ == „abc“

anwenden (return „abc“) (return „def“) == IO „abcdef“