Denken Sie daran, die unterfertigte eidesstattliche Erklärung mit abzugeben!

185.A03 Funktionale Programmierung WS 21

Schriftlicher Test 2 auf Papier

Freitag, 04.03.2022, 16:00–18:00 Uhr

Name	Vorname	Matr.-Nr.	Hörsaal	Platz	Unterschrift
			FH HS1		

Aufgabe	1	2	3	4	5	Summe Punkte
Punkte	16	18	27	21	18	100
Erreichte Punkte						

Aufgabe 1 (2∗0.5+8+3+4 = 16 Punkte)

Eine nichtleere Folge öffnender und schließender Klammern ist ein *Klammergebirge* genau dann, wenn die Zahlen öffnender und schließender Klammern übereinstimmen und an jeder Stelle in der Folge die Zahl öffnender Klammern mindestens so groß wie die schließender Klammern ist.

1. Richtig oder falsch?

   Die Klammerfolge

   (a) `(()(()))(())`
   (b) `((())()))(()`

   ist ein Klammergebirge (keine Begründung erforderlich).

2. Wir modellieren Klammergebirge über dem Datentyp `Klammer`, wobei das Schlüsselwort der Deklaration fehlt, `A` für eine öffnende ('Klammer auf') und `Z` für eine schließende Klammer ('Klammer zu') steht:

   ```
   <Schluesselwort zu ergaenzen> Klammer      = A | Z
   <Schluesselwort zu ergaenzen> Klammerfolge = [Klammer]
   ```

   Schreiben Sie eine Wahrheitswertfunktion `ist_kg`:

   ```
   ist_kg :: Klammerfolge -> Bool
   ```

   die angewendet auf einen `Klammerfolge`-Wert `True` liefert, wenn sie ein Klammergebirge ist, `False` sonst. Vervollständigen Sie dabei die Deklarationen von `Klammer` und `Klammerfolge` und nehmen Sie nötige Typklasseninstanziierungen vor.

3. Welche Fehlerbehandlungsweise ist durch die Aufgabenbeschreibung für `ist_kg` gefordert? Begründen Sie Ihre Antwort.

4. Erklären Sie knapp, aber gut nachvollziehbar, wie Ihre Implementierung vorgeht.

Denken Sie daran, die unterfertigte eidesstattliche Erklärung mit abzugeben!

Aufgabe 2 (2+8+4+2∗2 = 18 Punkte)

Zwei Listen sind *ähnlich* genau dann, wenn sie sich nur in der Reihenfolge ihrer Elemente unterscheiden.

1. Richtig oder falsch?

   Eine Implementierung von `sind_ae`, die angewendet auf zwei Listen bestimmt, ob sie ähnlich sind oder nicht, kann ausnutzen, dass ähnliche Listen nach Sortierung ident sind:

   ```
   sind_ae :: Eq a => [a] -> [a] -> Bool
   ```

   Begründen Sie Ihre Antwort.

2. Implementieren Sie `sind_ae` aus Aufgabe 2.1 direkt rekursiv. Geben Sie dabei auch die Implementierung und Typsignatur aller ggf. benutzter Hilfsfunktionen vollständig an.

3. Erklären Sie knapp, aber gut nachvollziehbar, wie Ihre Implementierung von `sind_ae` vorgeht.

4. Von welchem

   (a) Rekursionstyp

   (b) Polymorphietyp

   ist Ihre Implementierung von `sind_ae`? Geben Sie Rekursions- und Polymorphietyp möglichst genau an und begründen Sie Ihre Antwort.

Denken Sie daran, die unterfertigte eidesstattliche Erklärung mit abzugeben!

Aufgabe 3 (2+3+3+3∗1+3∗1+3∗3+4 = 27 Punkte)

Gegeben sind die unvollständigen Deklarationen der Typen `(T1 a)`, `(T2 a)`, `(T3 a)` und die Deklaration der Funktion `f`. Alle drei Typdeklarationen benutzen die Typvariable `a` auch rechtsseitig:

```
<Schluesselwort zu ergaenzen> T1 a = ...
<Schluesselwort zu ergaenzen> T2 a = ...
<Schluesselwort zu ergaenzen> T3 a = ...

f :: T1 a -> T2 a -> T3 a
f (F g) (B x y)   = [g x y]
f (F g) (K l m r) = (f (F g) l) ++ m ++ (f (F g) r)
```

1. Geben Sie den Rekursionstyp von `f` an. Begründen Sie Ihre Antwort.
2. Markieren Sie alle Konstruktorvorkommen in der Deklaration von `f` (z.B. durch Einkreisen).
3. Welche Stelligkeit haben diese Konstruktoren? Begründen Sie Ihre Antwort.
4. Wieviele und welche Konstruktoren haben die Typen
   (a) `T1 a`
   (b) `T2 a`
   (c) `T3 a`?
5. Geben Sie die Typen dieser Konstruktoren an und begründen Sie Ihre Antwort.
6. Vervollständigen Sie die Deklarationen von
   (a) `T1 a`
   (b) `T2 a`
   (c) `T3 a`

   so, dass die Deklaration von `f` typkorrekt ist und typkorrekte Aufrufe von `f` nicht zu *Musterfehlern**) führen. Begründen Sie Ihre Antwort.

   *) *Anm.:* Zu einem *Musterfehler* kommt es, wenn ein Argumentwert auf kein Muster der Implementierungszeilen einer Funktion passt; Musterfehler führen zu einem Auswertungsabbruch: *"program abortion by pattern exhaustion"*. Im Beispiell von `ordering2int` mit:

   ```
   ordering2int :: Ordering-> Int
   ordering2lt LT = -1
   ordering2lt EQ = 0
   ```

   führt der Aufruf `ordering2int GT` zu einem Musterfehler und Auswertungsabbruch.
7. Richtig oder falsch?

   Die Schlüsselwörter zur Deklaration von:
   (a) `T1 a`
   (b) `T2 a`
   (c) `T3 a`

   in Aufgabe 3.5 sind jeweils eindeutig bestimmt.

   Begründen Sie Ihre Antwort.

Denken Sie daran, die unterfertigte eidesstattliche Erklärung mit abzugeben!

Aufgabe 4 (3∗1+3∗1+3∗1+8+4 = 21 Punkte)

Ganze rationale Funktionen $f : \mathbb{R} \to \mathbb{R}$ auf der Menge $\mathbb{R}$ reeller Zahlen sind von der Form:

$$f(x) = \sum_{i=0}^{n} c_i x^i$$

Dabei gilt: $n \in \mathbb{N}_0$, $x \in \mathbb{R}$, $c_i \in \mathbb{R}$ für $i \in \{0, \ldots, n\}$ und $c_n = 0 \Rightarrow n = 0$. Mit f ist auch die Ableitung f' von f eine ganze rationale Funktion mit:

$$f'(x) = \sum_{i=1}^{n} i \cdot c_i x^{i-1}$$

darstellen.

1. Geben Sie die Ableitungen f_1', f_2', f_3' von f_1, f_2, f_3 an:

   (a) $f_1(x) = 3x^2 + 2x + 5$
   (b) $f_2(x) = 5x^4 + 0x^3 - 3x^2 + 2x + 0$
   (c) $f_3(x) = 3.14x^3 - 2.71x^1 + 1.57$

2. Ganze rationale Funktionen f mit $f(x) = \sum_{i=0}^{n} c_i x^i$ lassen sich kompakt durch die Liste ihrer Koeffizienten

   $$[c_n, \ldots, c_2, c_1, c_0]$$

   darstellen.

   Geben Sie die Koeffizientenlistendarstellungen von f_1, f_2, f_3 und ihrer Ableitungen f_1', f_2', f_3' an.

3. Wir modellieren ganze rationale Funktionen als Koeffizientenlisten:

   ```
   type Koeffizient = Double
   type RatFunktion = [Koeffizient]
   ```

   Richtig oder falsch?

   Die Listen

   (a) `[]`
   (b) `[0.0]`
   (c) `[0.0,0.0]`

   sind Darstellungen ganzer rationaler Funktionen. Geben Sie, wenn richtig, die ggf. von `[]`, `[0.0]`, `[0.0,0.0]` dargestellte Funktion an.

4. Schreiben Sie eine Haskell-Rechenvorschrift `abl`:

   ```
   abl :: RatFunktion -> RatFunktion
   ```

   die angewendet auf eine ganze rationale Funktion deren Ableitung berechnet. Dabei werden etwaige im Argument enthaltene führende Nullen (d.h. $\exists k \in \{1, \ldots, n\}.\ \forall j \geq k.\ c_j = 0$) vor einer Fehlerprüfung mit im Fehlerfall transparenter Fehlerbehandlung gestrichen.

5. Erklären Sie knapp, aber gut nachvollziehbar, wie Ihre Implementierung von `abl` vorgeht und warum die Fehlerbehandlung transparent ist.

Hinweis: `fromIntegral :: (Integral a,Num b) => a -> b` wandelt Werte verschiedener Zahltypen ineinander um, z.B. `Int`- in `Double`-Werte, und kann, wenn nötig, benutzt werden.

Denken Sie daran, die unterfertigte eidesstattliche Erklärung mit abzugeben!

Aufgabe 5 (3∗1+5+5+2∗2.5 = 18 Punkte)

Statt durch Koeffizientenlisten lassen sich die in Aufgabe 4 eingeführten ganzen rationalen Funktionen auch auf folgende Weise modellieren:

```
type Koeffizientenindex    = Int
type MaxKoeffizientenindex = Koeffizientenindex
data RatFunktion'          = RF MaxKoeffizientenindex (Koeffizientenindex -> Double)
```

1. Welche `RatFunktion'`-Werte entsprechen den `RatFunktion`-Werten:

   (a) `[]`

   (b) `[0.0]`

   (c) `[3.14,2.71,1.57]`?

   Geben Sie entsprechende Werte an und begründen Sie Ihre Antwort.

2. Schreiben Sie eine Transformationsfunktion `t`, die `RatFunktion`-Werte in entsprechende `RatFunktion'`-Werte transformiert:

   ```
   t :: RatFunktion -> RatFunktion'
   ```

3. Schreiben Sie eine Transformationsfunktion `t'`, die `RatFunktion'`-Werte in entsprechende `RatFunktion`-Werte transformiert:

   ```
   t' :: RatFunktion' -> RatFunktion
   ```

4. Erklären Sie knapp, aber gut nachvollziehbar, wie Ihre Implementierungen von

   (a) `t`

   (b) `t'`

   vorgehen.