Eine Sache lernt man, indem man sie macht.
Cesare Pavese (1908-1950)
italien. Schriftsteller

Für das Können gibt es nur einen Beweis, das Tun.
Marie von Ebner-Eschenbach (1830-1916)
österr. Schriftstellerin

2. Angabe zu Funktionale Programmierung von Fr, 21.10.2022

Erstabgabe: Fr, 28.10.2022, 12:00 Uhr

Zweitabgabe: Siehe „Hinweise zu Org. u. Ablauf der Übung" (TUWEL-Kurs)

(Teil A: beurteilt; Teil B: ohne Abgabe, ohne Beurteilung)

Themen: *Typaliase, neue Typen, algebraische Typen, curryfizierte Funktionen, automatische Operatorüberladung, Muster*

Stoffumfang: *Kapitel 1 bis Kapitel 6.*

- **Teil A, programmiertechnische Aufgaben:** Besprechung am ersten Übungsgruppentermin, der auf die *Zweitabgabe* der programmiertechnischen Aufgaben folgt.
- **Teil B, Papier- und Bleistiftaufgaben:** Besprechung am ersten Übungsgruppentermin, der auf die *Erstabgabe* der programmiertechnischen Aufgaben folgt.
- **Teil C, das Sprachduell:** eine Aufgabe besser für Haskell oder für eine andere Sprache?

Wichtig

1. Befolgen Sie die Anweisungen aus den 'Lies-mich'-Dateien (s. TUWEL-Kurs) zu den Angaben sorgfältig, um ein reibungsloses Zusammenspiel mit dem Testsystem sicherzustellen. Bei Fragen dazu, stellen Sie diese bitte im TUWEL-Forum zur LVA.
2. Erweitern Sie für die für diese Angabe zu schreibenden Rechenvorschriften die zur Verfügung gestellte Rahmendatei

   `Angabe2.hs`

   und legen Sie sie für die Abgabe auf oberstem Niveau in Ihrem *home*-Verzeichnis ab. Achten Sie darauf, dass "Gruppe" Leserechte für diese Datei hat. Wenn nicht, setzen Sie diese Leserechte mittels `chmod g+r Angabe2.hs`.

   Löschen Sie keinesfalls eine Deklaration aus der Rahmendatei! Auch dann nicht, wenn Sie einige dieser Deklarationen nicht oder nicht vollständig implementieren wollen. Löschen Sie auch nicht die für das Testsystem erforderliche Modul-Anweisung `module Angabe2 where` am Anfang der Rahmendatei.
3. Der Name der Abgabedatei ist für Erst- und Zweitabgabe ident!
4. Benutzen Sie keine selbstdefinierten Module! Wenn Sie (für spätere Angaben) einzelne Rechenvorschriften früherer Lösungen wiederverwenden möchten, kopieren Sie diese bitte in die neue Abgabedatei ein. Eine `import`-Anweisung für selbstdefinierte Module schlägt für die Auswertung durch das Abgabesystem fehl, weil Ihre Modul-Datei, aus der importiert werden soll, vom Testsystem nicht mit abgesammelt wird.
5. Ihre Programmierlösungen werden stets auf der Maschine `g0` mit der dort installierten Version von GHCi überprüft. Stellen Sie deshalb sicher, dass sich Ihre Programme (auch) auf der `g0` unter GHCi so verhalten, wie von Ihnen gewünscht.

6. Überzeugen Sie sich bei jeder Abgabe davon! Das gilt besonders, wenn Sie für die Entwicklung Ihrer Haskell-Programme mit einer anderen Maschine, einer anderen GHCi-Version oder/und einem anderen Werkzeug wie etwa Hugs arbeiten!

A Programmiertechnische Aufgaben (beurteilt, max. 50 Punkte)

Erweitern Sie zur Lösung der programmiertechnischen Aufgaben die Rahmendatei `Angabe2.hs`. Kommentieren Sie die Rechenvorschriften in Ihrem Programm zweckmäßig, aussagekräftig und problemangemessen. Benutzen Sie, wo sinnvoll, Hilfsfunktionen und Wertvereinbarungen für konstante Werte (z.B. `pi = 3.14 :: Float`). Versehen Sie alle Funktionen, die Sie zur Lösung der Aufgaben benötigen, mit ihren Typdeklarationen, d.h. geben Sie stets deren syntaktische Signatur (kurz: Signatur), explizit an.

Wir führen folgende Datentypen ein:

```
data Lotterielos = Treffer
                 | Niete
                 | Freilos deriving (Eq,Show)

data Liste = Schluss Lotterielos
           | Kopf Lotterielos Liste deriving (Eq,Show)

data Baum = Blatt Lotterielos
          | Gabel Baum Lotterielos Baum deriving (Eq,Show)

data Liste' = Schluss' Baum
            | Kopf' Baum Liste' deriving (Eq,Show)

data Baum' = Blatt' Liste
           | Gabel' Baum' Liste Baum' deriving (Eq,Show)

type Loswert    = Lotterielos
type Auswertung = Ordering
```

A.1 Schreiben Sie eine Haskell-Rechenvorschrift `analysiere` mit Signatur:

```
analysiere :: Liste -> Loswert -> Loswert -> Auswertung
```

Angewendet auf eine Liste l und zwei Loswerte lw und lw', berechnet `analysiere`, ob lw öfter, gleich oft oder weniger oft als lw' in l vorkommt und gibt entsprechend den Wert `GT` für öfter, `EQ` für gleich oft und `LT` für weniger oft zurück.

A.2 Schreiben Sie eine Haskell-Rechenvorschrift `analysiere'` mit Signatur:

```
analysiere' :: Baum -> Loswert -> Loswert -> Auswertung
```

Angewendet auf einen Baum b und zwei Loswerte lw und lw', berechnet `analysiere'`, ob lw öfter, gleich oft oder weniger oft als lw' in b vorkommt und gibt entsprechend den Wert `GT` für öfter, `EQ` für gleich oft und `LT` für weniger oft zurück.

A.3 Schreiben Sie eine Haskell-Rechenvorschrift `analysiere''` mit Signatur:

```
analysiere'' :: Liste' -> Loswert -> Loswert -> Auswertung
```

Angewendet auf ein Liste l und zwei Loswerte lw und lw', berechnet `analysiere''` ob lw öfter, gleich oft oder weniger oft als lw' in b vorkommt und gibt entsprechend den Wert `GT` für öfter, `EQ` für gleich oft und `LT` für weniger oft zurück.

A.4 Schreiben Sie eine Haskell-Rechenvorschrift `analysiere'''` mit Signatur:

```
analysiere''' :: Baum' -> Loswert -> Loswert -> Auswertung
```

Angewendet auf einen Baum b und zwei Loswerte lw und lw', berechnet `analysiere'''`, ob lw öfter, gleich oft oder weniger oft als lw' in b vorkommt und gibt entsprechend den Wert `GT` für öfter, `EQ` für gleich oft und `LT` für weniger oft zurück.

A.5 **Ohne Beurteilung:** Beschreiben Sie für jede Rechenvorschrift in einem Kommentar knapp, aber gut nachvollziehbar, wie die Rechenvorschrift vorgeht.

A.6 **Ohne Abgabe, ohne Beurteilung:** Testen Sie alle Funktionen umfassend mit aussagekräftigen eigenen Testdaten.

B Papier- & Bleistiftaufgaben (ohne Abgabe, ohne Beurteilung)

B.1 (a) Welche Typen in den Aufgaben A.1 bis A.4 sind algebraische Typen, welche nicht?

(b) Woran erkennt man das jeweils?

B.2 Welche der Typen in den Aufgaben A.1 bis A.4 sind

(a) Aufzählungstypen?

(b) Summentypen?

(c) Produkttypen?

Woran erkennt man das jeweils?

B.3 Lassen sich einige der `data`-Deklarationen n den Aufgaben A.1 bis A.4 durch `newtype`-Deklarationen ersetzen? Wenn ja, warum? Wenn nein, warum nicht?

B.4 Die Typsignaturen der Funktionen den Aufgaben A.1 bis A.4 sind klammerfrei angegeben.

(a) Warum geht das?

(b) Geben Sie die vollständig, aber nicht überflüssig geklammerten Signaturen an.

B.5 (a) Welchen Typ haben die Funktionen `fac`, `fib` und `fab` in folgendem Beispiel?

```
fehlerwert = (-1) :: Int

fac n =
 | n == 0 = 1
 | n > 0  = n * fac (n-1)
 | n < 0  = fehlerwert

fib n =
 | n == 0 = 0
```

```
  | n == 1 = 1
  | n > 1  = fib (n-1) + fib (n-2)
  | n < 0  = fehlerwert

fab n
  | n >= 0 = fac
  | n < 0  = fib
```

(b) Wie ändern sich die Typen der Funktionen, wenn die Deklaration von `fehlerwert` wie folgt geändert wird?

i. `fehlerwert = (-1) :: Integer`

ii. `fehlerwert = (-1)`

Überprüfen Sie Ihre Vermutungen mithilfe von `ghci` oder/und `hugs`! Sind die Klammern um "-1" nötig?

B.6 Welches Ergebnis liefern folgende Aufrufe?

(a) `fab 42 5`

(b) `fab (-42) 5`

Überprüfen Sie Ihre Vermutungen mithilfe von `ghci` oder/und `hugs`!

B.7 Die Ausdrücke in B.6 sind klammerfrei angegeben.

(a) Warum geht das?

(b) Geben Sie die Ausdrücke aus B.6 vollständig, aber nicht überflüssig geklammert an.

B.8 Sind durch `Loswert` und `Auswertung` eigenständige Typen eingeführt? Begründen Sie Ihre Antwort.

Iucundi acti labores.
Getane Arbeiten sind angenehm.

Cicero (106 - 43 v.Chr.)
röm. Staatsmann und Schriftsteller

C Das Sprachduell: eine Aufgabe besser für Haskell oder für eine andere Sprache?

Krypto Kracker!

Eine gleichermaßen populäre wie einfache und unsichere Methode zur Verschlüsselung von Texten besteht darin, eine Permutation des Alphabets zu verwenden. Bei dieser Methode wird jeder Buchstabe des Alphabets einheitlich durch einen anderen Buchstaben ersetzt, wobei keine zwei Buchstaben durch denselben Buchstaben ersetzt werden. Das stellt sicher, dass verschlüsselte Texte auch wieder eindeutig entschlüsselt werden können.

Eine Standardmethode zur Entschlüsselung nach obiger Methode verschlüsselter Texte ist als "reiner Textangriff" bekannt. Diese Angriffsmethode beruht darauf, dass der Angreifer den Klartext einer Textphrase kennt, von der er weiß, dass sie in verschlüsselter Form im Geheimtext vorkommt. Durch den Vergleich von Klartext- und verschlüsselter Phrase wird auf die Verschlüsselung geschlossen, d.h. auf die verwendete Permutation des Alphabets. In unserem Fall wissen wir, dass der Geheimtext die Verschlüsselung der Klartextphrase

```
the quick brown fox jumps over the lazy dog
```

enthält.

Ihre Aufgabe ist nun, eine Liste von Geheimtextphrasen, von denen eine die obige Klartextphrase codiert, zu entschlüsseln und die entsprechenden Klartextphrasen auszugeben. Kommt mehr als eine Geheimtextphrase als Verschlüsselung obiger Klartextphrase in Frage, geben Sie alle möglichen Entschlüsselungen der Geheimtextphrasen an. Im Geheimtext kommen dabei neben Leerzeichen ausschließlich Kleinbuchstaben vor, also weder Ziffern noch sonstige Sonderzeichen.

Schreiben Sie ein Programm, das diese Entschlüsselung für eine Liste von Geheimtextphrasen vornimmt. In einer Sprache Ihrer Wahl!

C.1 Treffen Sie Ihre Wahl für das Sprachduell! (Wenn es nicht Haskell ist, kommen Sie am Ende des Semesters noch einmal auf das Duell zurück!)

C.2 Schreiben Sie in der Sprache, die Sie gewählt haben, ein Programm, das die Entschlüsselung vornimmt.

Angewendet auf den aus drei Geheimtextphrasen bestehenden Geheimtext (der in Form einer Haskell-Liste von Zeichenreihen vorliegt)

```
["vtz ud xnm xugm itr pyy jttk gmv xt otgm xt xnm puk ti xnm fprxq",
 "xnm ceuob lrtzv ita hegfd tsmr xnm ypwq ktj",
 "frtjrpgguvj otvxmdxd prm iev prmvx xnmq"]
```

sollte Ihre Entschlüsselungsprogramm folgende Klartextphrasen liefern (ebenfalls wieder in Form einer Haskell-Liste von Zeichenreihen):

```
["now is the time for all good men to come to the aid of the party",
 "the quick brown fox jumps over the lazy dog",
 "programming contests are fun arent they"]
```

(*Lösungsvorschläge für das Sprachduell werden im Rahmen ausreichender Zeit in einer der Plenumsübungen besprochen.*)