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

4. Angabe zu Funktionale Programmierung von Fr, 04.11.2022.

Erstabgabe: Fr, 11.11.2022, 12:00 Uhr

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

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

Themen: *Literate Haskell-Skripte, Feldsyntax algebraischer und neuer Datentypen, Rechnen mit Listen und Tupeln, musterbasierte Fallunterscheidungen in Funktionsdefinitionen, Umformen und umrechnen von Werten verschiedener Typen*

Stoffumfang: *Kapitel 1 bis Kapitel 10, besonders Kapitel 2 bis 6 und 10.*

- **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

   `Angabe4.lhs`

   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 Angabe4.lhs`.

   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 Angabe4 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 `Angabe4.lhs`. Kommentieren Sie die Rechenvorschriften in Ihrem Programm zweckmäßig, aussagekräftig und problemangemessen. Benutzen Sie, wo sinnvoll, Hilfsfunktionen und Konstanten. Versehen Sie alle Funktionen, die Sie zur Lösung der Aufgaben benötigen, mit ihren Typdeklarationen; geben Sie also stets deren syntaktische Signatur (kurz: Signatur), explizit an.

Wir betrachten ein einfaches Suchmaschinenproblem für den günstigen Einkauf von Waschmaschinen, Wäschetrocknern und Wäscheschleudern:

```
> type Nat0    = Int     -- Natürliche Zahlen beginnend mit 0
> type Nat1    = Int     -- Natürliche Zahlen beginnend mit 1
> type Nat2023 = Int     -- Natürliche Zahlen beginnend mit 2023

> newtype EUR  = EUR { euro :: Nat1 }

> data Skonto  = Kein_Skonto
>              | DreiProzent
>              | FuenfProzent
>              | ZehnProzent

> data Waschmaschinentyp   = WM_Typ1 | WM_Typ2 | WM_Typ3 | WM_Typ4 | WM_Typ5
> data Waeschetrocknertyp  = WT_Typ1 | WT_Typ2 | WT_Typ3 | WT_Typ4
> data Waescheschleudertyp = WS_Typ1 | WS_Typ2 | WS_Typ3

> data Typ = WM Waschmaschinentyp
>          | WT Waeschetrocknertyp
>          | WS Waescheschleudertyp

> data Quartal       = Q1 | Q2 | Q3 | Q4 deriving (Eq,Ord,Show)
> type Jahr          = Nat2023
> data Lieferfenster = LF { quartal :: Quartal,
>                           jahr    :: Jahr
>                         }

> data Datensatz
>   = DS { preis_in_euro :: Nat1,
>          sofort_lieferbare_stueckzahl :: Nat0,
>          lieferbare_stueckzahl_im_Zeitfenster :: Lieferfenster -> Nat0,
>          skonto :: Skonto
>        }
>     | Nicht_im_Sortiment
```

```
> data Sortiment
>   = WMS {wm   :: Waschmaschinentyp   -> Datensatz}
>     | WTS {wt :: Waeschetrocknertyp  -> Datensatz}
>     | WSS {ws :: Waescheschleudertyp -> Datensatz}

> data Lieferantenname = L1 | L2 | L3 | L4 | L5 | L6 | L7 | L8 | L9 | L10

> type Lieferanten = Lieferantenname -> Sortiment

> type Suchanfrage = Typ
```

A.1 Welche Lieferanten können sofort liefern?

```
> type Lieferantenliste = [Lieferantenname]

> sofort_erhaeltlich_bei :: Suchanfrage -> Lieferanten
>                                                -> Lieferantenliste
```

Kann mehr als ein Lieferant sofort liefern, soll die Ergebnisliste aufsteigend sortiert sein (dabei gilt: `L1 < L2 < L3 < ... < L10`); kann kein Lieferant sofort liefern, bleibt die Ergebnisliste leer.

A.2 Wieviel Stück sind sofort erhältlich, wenn bei allen Lieferanten die jeweils maximal mögliche sofort lieferbare Stückzahl bestellt wird? Wie hoch ist der zugehörige unskontierte Gesamtpreis?

```
> type Stueckzahl  = Nat0
> type Gesamtpreis = Nat0

> sofort_erhaeltliche_Stueckzahl :: Suchanfrage -> Lieferanten
>                                          -> (Stueckzahl,Gesamtpreis)
```

Kann kein Lieferant sofort liefern, weist das Ergebnis jeweils 0 für Stückzahl und Gesamtpreis aus.

A.3 Welche Lieferanten können im angegebenen Lieferfenster ohne Berücksichtigung eines möglichen Skontorabatts am günstigsten liefern?

```
> type Preis = EUR
> guenstigste_Lieferanten :: Suchanfrage -> Lieferfenster -> Lieferanten
>                                              -> Maybe Lieferantenliste
```

Kann mehr als ein Lieferant im angegebenen Lieferfenster günstigst liefern, soll die Ergebnisliste aufsteigend sortiert (s. A.1) als `Just`-Wert des Typs `Maybe` geliefert werden; kann kein Lieferant im angegebenen Fenster liefern, wird als Ergebnis der `Maybe`-Wert `Nothing` ausgegeben.

A.4 Welche Lieferanten können im angegebenen Lieferfenster mindestens die vorgegebene Stückzahl unter Berücksichtigung eines möglichen Skontorabatts am günstigsten liefern? Der rabattierte Preis wird dabei vom unskontierten Gesamtbetrag berechnet und auf den nächsten vollen Eurobetrag aufgerundet.

```
> type RabattierterPreis = EUR

> guenstigste_Lieferanten_im_Lieferfenster ::
>   Suchanfrage -> Lieferfenster -> Stueckzahl -> Lieferanten
>     -> [(Lieferantenname,RabattierterPreis)]
```

Kann mehr als ein Lieferant im angegebenen Lieferfenster die benötigte Stückzahl günstigst liefern, soll die Ergebnisliste aufsteigend nach Lieferantennamen sortiert sein (s. A.1); kann kein Lieferant mindestens die benötigte Stückzahl im angegebenen Zeitfenster liefern, wird als Ergebnis die leere Liste geliefert.

A.5 Ergänzen Sie in den Typdeklarationen dort, wo nötig, `deriving`-Klauseln, so dass nötige Vergleichstests auf Werten von Typen ausgeführt werden können und alle Ergebnisse am Bildschirm ausgegeben werden können. Dafür sind die Typklassen `Eq`, `Ord`, `Show`, ggf. auch `Enum` wichtig.

A.6 **Ohne Beurteilung:** Beschreiben Sie für jede Rechenvorschrift in einem Kommentar knapp, aber gut nachvollziehbar, wie die Rechenvorschrift vorgeht.

A.7 **Ohne Abgabe, ohne Beurteilung:** Testen Sie alle Funktionen umfassend mit aussagekräftigen eigenen Testdaten.

B Papier- und Bleistiftaufgaben (ohne Abgabe/Beurteilung)

B.1 Welchen Rekursionstyp haben die Funktionen, die sie für Angabe 4 rekursiv implementiert haben? Woran haben Sie die Rekursionstypen jeweils erkannt? Welche Rekursionstypen aus Kapitel 7.2 sind nicht vorgekommen (falls es solche gibt)?

B.2 Geben Sie die Aufrufgraphen der Funktionen an, die Sie für die Lösung von Angabe 4 geschrieben haben (s. Kapitel 7.3).

B.3 Wir ändern den Datentyp `Sortiment` in folgender Weise ab:

```
> newtype Sortiment = S { artikel :: Typ -> Datensatz }
```

B.3.1 Leistet ein Lieferant in beiden Modellierungen dasselbe oder ändert sich die 'äußere' Semantik, die Sicht auf Lieferanten durch die Änderung?

B.3.2 Wie können Sie Ihre Implementierung auf den neuen Typ von `Sortiment` anpassen?

B.3.3 Wird die Implementierung durch die Änderung aufwändiger, weniger aufwändig?

Begründen Sie Ihre Antworten.

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? (ohne Abgabe/Beurteilung)

Welt perfekter Türme – World of Perfect Towers

In diesem Spiel konstruieren wir auf einem Spielfeld *Welten perfekter Türme*.

Das Spielfeld besteht dabei aus einer Bodenplatte, auf der n Stäbe senkrecht stehend befestigt sind. Auf jeden dieser Stäbe können Kugeln gesteckt werden, die dafür mit einer entsprechenden Bohrung versehen sind. Sowohl die Kugeln wie die Stäbe sind beginnend mit jeweils 1 fortlaufend nummeriert.

Die auf einen Stab gesteckten Kugeln bilden einen Turm. Die zuerst aufgesteckte Kugel liegt dabei direkt auf der Bodenplatte ganz unten im Turm, die zu zweit aufgesteckte Kugel liegt auf der zuerst aufgesteckten, usw.; die zuletzt aufgesteckte Kugel liegt ganz oben im Turm. Ein solcher Turm heißt *perfekt*, wenn die Summe der Nummern zweier unmittelbar übereinanderliegender Kugeln eine Zweierpotenz ist. Eine Menge von n perfekten Türmen heißt *n-perfekte Welt*.

In diesem Spiel geht es darum, n-perfekte Welten mit maximaler Kugelzahl zu konstruieren. Dazu werden beginnend mit der mit 1 nummerierten Kugel die Kugeln in aufsteigender Nummerierung so auf die n Stäbe der Spielwelt gesteckt, dass die Kugeln auf jedem Stab einen perfekten Turm bilden und die Summe der Kugeln aller Türme maximal ist.

Schreiben Sie ein Programm, das zu einer vorgegebenen Zahl n von Stäben die Maximalzahl von Kugeln einer n-perfekten Welt bestimmt und die Türme dieser n-perfekten Welt in Form einer Liste von Listen ausgibt, wobei jede Liste von links nach rechts die Kugeln des zugehörigen Turms in aufsteigender Reihenfolge angibt.

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 Aufgabe löst und die Lösung in der beschriebenen Form ausgibt.

(*Lösungsvorschläge für das Sprachduell werden in einer der kommenden Plenumsübungen besprochen, wenn es die Zeit erlaubt.*)