# Algorithmieren und Programmieren

Sommersemester 2023

*Lehrstuhl Programmiersprachen und Compilerbau*
*Prof. Dr. rer. nat. habil. Petra Hofstedt*
*Ilja Becker*
*Andreas Eberle, Jan Robert Meyer, Oliver Pflug*

Brandenburgische
Technische Universität
Cottbus - Senftenberg

## Übungsblatt 3

Abgabedatum: 07.05.2023

### Hinweise

- Beachten Sie, dass die Tutoren unkommentierte Programme nicht als Lösung akzeptieren, selbst wenn die Programme richtig funktionieren. Zu einer richtigen Lösung gehören immer sinnvolle Kommentare, deren Umfang der Komplexität des Programms angemessen ist.
- **Halten Sie sich an die in den Übungsblättern vorgegebenen Namen von Funktionen und Dateien, Funktionstypen (Typsignaturen), Reihenfolge der Parameter und verwenden Sie — sofern vorhanden — die Vorgaben!**
- Auf den Übungsblättern finden Sie einige Haskell-Quelltextfragmente. Diese sind, der besseren Lesbarkeit wegen, unter Nutzung einiger mathematischer Sonderzeichen wiedergegeben.
- Für diese Veranstaltung wird die Verwendung des ***Glasgow Haskell Compiler (GHC)*** (`https://www.haskell.org/`) empfohlen. Weitere Informationen finden Sie im Software-Reiter auf Moodle.
- Als **Tutoriumsaufgabe** markierte (Teil-)aufgaben werden in den Übungen ausführlicher besprochen. Die schriftlichen bzw. elektronischen Lösungen müssen jedoch trotzdem mit abgegeben werden. Bitte schauen Sie Sich diese Aufgaben im Vorfeld der Übung an und bereiten Sie Sich darauf vor.
- Die Abgabe Ihrer Lösungen erfolgt vor Ablauf der Abgabefrist digital über die Moodle-Plattform an Ihren Tutor. Erstellen Sie dazu ein PDF-Dokument, das die Lösungen Ihrer schriftlichen Aufgaben enthält. Laden Sie dieses PDF-Dokument und die erzeugten Hakell-Dateien, mit den in den Aufgaben vorgegebenen Namen, bei Moodle hoch.

### Informationsquellen

- Sie finden unter `http://haskell.org/` sehr viele Informationen über die Programmiersprache Haskell. Von besonderem Interesse sind dabei sicherlich die Übersicht über zahlreiche online verfügbare Haskell-Tutorials (`https://wiki.haskell.org/Tutorials`) sowie die Suchmaschine Hoogle (`http://hoogle.haskell.org/`) für die Haskell-API, die Ihnen mit zunehmender Haskell-Erfahrung wertvolle Dienste leisten wird.

Sie können maximal **(6 Punkte)** mit diesem Übungsblatt erreichen.

**Aufgabe 1 (Funktionen)** 1 Punkt

Sei $S$ die Menge aller Studierenden der BTU Cottbus, $F$ die Menge der quadratischen Funktionen. Die Tripel $f_i = (D_i, W_i, R_i)$ mit $i \in \{1, \ldots, 5\}$ seien wie folgt definiert:

| | | |
|---|---|---|
| $D_1 = \mathbb{N} \times \mathbb{N}$ | $W_1 = \mathbb{N}$ | $R_1 = \{((a,b),c) \mid a + b = c\}$ |
| $D_2 = \mathbb{N} \times \mathbb{N} \times \mathbb{N}$ | $W_2 = \mathbb{N} \times \mathbb{N} \times \mathbb{N}$ | $R_2 = \{((a,b,c),(d,e,f)) \mid \forall x \in \mathbb{N} : ax^2 + bx + c = dx^2 + ex + f\}$ |
| $D_3 = F$ | $W_3 = \mathbb{R}$ | $R_3 = \{(f,b) \mid f(b) = 0\}$ |
| $D_4 = S$ | $W_4 = S$ | $R_4 = \{(a,b) \mid a \text{ kennt } b\}$ |
| $D_5 = S$ | $W_5 = \mathcal{P}(S)$ | $R_5 = \{(a,B) \mid B = \{b \mid b \in S \wedge a \text{ kennt } b\}\}$ |

Welche dieser Tripel sind Funktionen? Begründen Sie Ihre Antworten.

**Aufgabe 2 (Datentypen)** 2 Punkte

1. **(Tutoriumsaufgabe)** Was sind Datentypen? Welche Mengen werden durch Datentypen charakterisiert? Betrachten Sie hierbei Produkt-, Summen- und Aufzählungstypen.

2. **Verwenden Sie für die Abgabe den Dateinamen:** `Datentypen.hs`

   In der Vorlesung haben Sie bereits einen Datentyp zur näherungsweisen Repräsentation von Punkten im Raum $\mathbb{R}^2$ kennengelernt.

   ```
   data Punkt =Punkt {xKoord :: Double, yKoord :: Double}
   ```

   Diese Notationsweise ist auch als "Record"-Schreibweise bekannt[1].

   a) Geben Sie für die folgenden Objekte die Definition eines passenden Datentyps in der Record-Schreibweise an. Vergeben Sie dabei sinnvolle Namen für die Datenkonstruktoren und Selektoren.

   **(Tutoriumsaufgabe) Bruch** bestehend aus Zähler und Nenner

   **(Tutoriumsaufgabe) Bewertung** einer Hausaufgabe, die *Bestanden*, *Nichtbestanden* oder *Nachzubearbeiten* ist

   **Wochentag** entweder Montag, Dienstag, ... oder Sonntag

   **Monat** entweder Januar, Februar, ... oder Dezember

   **Datum** bestehend aus Tag, Monat und Jahr

   **Uhrzeit** bestehend aus Stunden, Minuten und Sekunden

   **Preis** bestehend aus Euro und Cent

   **Kasse** bestehen aus KassenID, und Nachname des Kassierers

   **Mensaessen** entweder Tagessuppe, Essen1, Essen2, Bioessen, Vegetarisch oder Aktionsessen

   **Umzugskarton** bestehend aus Volumen und Maximalgewicht

   **Kassenbon** enthält Kasse, Datum, `Mensaessen` und `Preis`

   b) Geben Sie für die eben Definitierten Typen jeweils an, ob es sich um einen Produkttyp, einen Summentyp oder um einen Aufzählungstyp handelt.

[1] s. `https://www.haskell.org/onlinereport/haskell2010/haskellch4.html#x10-680004.2` oder `https://devtut.github.io/haskell/record-syntax.html#basic-syntax`

c) Geben Sie jeweils den Typ des Datenkonstruktors an.

d) Geben Sie jeweils den Typ der automatisch generierten Selektor-Funktionen an.

e) Legen Sie einen Kassenbon an.

f) Ausgehend von einem Kassenbon, geben Sie für die folgenden Informationen eine geeignete Selektorfunktion sowie deren Funktionstyp an.

- **(Tutoriumsaufgabe)** Welcher Kassierer saß an der Kasse? Verwenden Sie den Funktionsnamen `kassiererVonKassenbon`.
- In welchem Jahr wurde der Kassenbon ausgestellt? Verwenden Sie den Funktionsnamen `jahrVomKassenbon`.
- Wie viel wurde bezahlt? Verwenden Sie den Funktionsnamen `preisVonKassenbon`.
- An welchem Tag wurde der Kassenbon ausgestellt? Verwenden Sie den Funktionsnamen `tagVonKassenbon`.

g) **Verwenden Sie für die Abgabe den Dateinamen:** `Datentypen-alternativ.hs`

Neben der sogenannten Record-Schreibweise kann für den Datentyp Punkt auch die folgende Schreibweise verwendet werden.

```
data Punkt =Punkt Double Double
```

Dabei werden Selektor-Funktionen nicht automatisch generiert.

Geben Sie für die Datentypen *Bruch*, *Datum*, *Uhrzeit*, *Kasse*, *Mensaessen* und *Kassenbon* eine entsprechende Typdefinition an.

Geben Sie jeweils eine Definition für die Selektor-Funktionen an.

### Aufgabe 3 (Pattern-Matching) 1 Punkt

Gegeben sind die folgenden Typdeklarationen:

```
data T = T String Integer deriving (Show)
data TListe = Leer | NichtLeer T TListe deriving (Show)
data VielleichtT = Nicht | Doch T deriving (Show)
```

1. Bestimmen Sie die Typen der Variablen (a - o) in den folgenden Mustern. Begründen Sie Ihre Antworten kurz.

```
loesche :: T → TListe → TListe
loesche a Leer = error "nicht da"
loesche (T b c) (NichtLeer (T d e) f )
    | (b == d) && (c == e) = f
    | otherwise = NichtLeer (T d e) (loesche (T b c) f )

ersetze :: T → TListe → TListe
ersetze g Leer = Leer
ersetze h@(T i _) (NichtLeer j@(T k _) l) = if i == k
    then NichtLeer h (ersetze h l)
    else NichtLeer j (ersetze h l)

findeAnIdx :: Integer → TListe → VielleichtT
findeAnIdx _ Leer = Nicht
findeAnIdx m (NichtLeer n o)
```

```
    | m == 0 = Doch n
    | otherwise = findeAnIdx ( pred m) o
```

2. Gegeben sind folgende Definitionen:

```
liste1 = NichtLeer (T "Ernie" 7)
            (NichtLeer (T "Bert" 9)
                (NichtLeer (T "Ernie" 7) Leer))

liste2 = NichtLeer (T "Ernie" 8)
            (NichtLeer (T "Bibo" 7) Leer)

liste3 = NichtLeer (T "Bibo" 8)
            (NichtLeer (T "Bibo" 7) Leer)
```

   Geben Sie die Resultate folgender Funktionsanwendungen an. Begründen Sie Ihre Antworten kurz.

```
loesche (T "Ernie" 7) liste1
ersetze (T "Bibo" 9) liste2
ersetze (T "Bibo" 9) liste3
findeAnIdx 1 liste3
```

**Aufgabe 4 (Geometrische Funktionen)** 2 Punkte

**Verwenden Sie für die Abgabe den Dateinamen:** `Geometrie.hs`

Eine Gerade kann in ihrer kartesischen Normalform durch $y = mx + n$ dargestellt werden.

1. **(Tutoriumsaufgabe)** Geben Sie einen geeigneten Datentyp zur Repräsentation von Geraden in der kartesischen Normalform an.

2. Programmieren Sie eine Haskellfunktion

   ```
   auswerten :: Gerade → Double → Double
   ```

   , welche eine Gerade und ein Argument (x-Wert) als Eingabe erhält und das zugehörige Ergebnis (y-Wert) zurückgibt.

3. Programmieren Sie eine Haskellfunktion

   ```
   schnittpunkt :: Gerade → Gerade → Punkt
   ```

   , welche den Schnittpunkt, sofern vorhanden, zweier Geraden berechnet.

4. Programmieren Sie eine Haskellfunktion

   ```
   flaecheZwischenGeraden :: Gerade → Gerade → Double → Double → Double
   ```

   Die Funktion soll die Fläche zwischen zwei Geraden ($f$ und $g$) im angegebenen Intervall ($[a, b]$) berechnen. Damit Sie Ihre Schnittpunkt-Funktion verwenden können, können Sie davon ausgehen, dass die beiden Geraden $f$ und $g$ über einen unterschiedlichen Anstieg verfügen.

   ```
   flaecheZwischenGeraden (Gerade (-0.5) 3) (Gerade 0.5 0) 0 6 ⇝ * 9.0
   ```