Algorithmieren und Programmieren

Sommersemester 2019

Prof. Dr. rer. nat. habil. Petra Hofstedt
Sven Löffler M. Sc.
Sonja Breuß, Deborah Buckenhauer, Johannes Kuhn, Julius Schöning, Carlo Bückert

Brandenburgische
Technische Universität
Cottbus - Senftenberg

Übungsblatt 2

Abgabedatum: 24.04.2019

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 der ***Haskell-Plattform*** (`https://www.haskell.org/platform/`) empfohlen.
- Als **Tutoriumsaufgabe** markierte (Teil-)aufgaben werden in den Übungen ausführlicher besprochen. Die schriftliche 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-Datein, 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 (`http://haskell.org/haskellwiki/Tutorials`) sowie die Suchmaschine Hoogle (`http://haskell.org/hoogle/`) für die Haskell-API, die Ihnen mit zunehmender Haskell-Erfahrung wertvolle Dienste leisten wird.
- Die Typen *Int* und *Integer* erlauben die folgenden Operationen:

Bezeichner	Typ	Bedeutung
succ	$a \to a$	Nachfolgerbildung
pred	$a \to a$	Vorgängerbildung
div	$a \to a \to a$	ganzzahlige Division, Ergebnis wird abgerundet
mod	$a \to a \to a$	zur ganzzahligen Division *div* gehörender Rest
quot	$a \to a \to a$	ganzzahlige Division, Ergebnis wird Richtung 0 gerundet
rem	$a \to a \to a$	zur ganzzahligen Division *quot* gehörender Rest

Sie können maximal **(4 Punkte)** mit diesem Übungsblatt erreichen.

Aufgabe 1 (bedingte Ausdrücke) 1 Punkt

Verwenden Sie für die Abgabe den Dateinamen: `Schaltjahr.hs`

Schreiben Sie folgende Funktionen vom Typ *Integer* → *Bool*, die zu einer gegebenen Jahreszahl prüfen, ob sie im Gregorianischen Kalender ein Schaltjahr ist oder nicht.

1. Funktion `schaltjahrIf`: Verwenden Sie hierfür nur $if - then - else$ -Ausdrücke.
2. Funktion `schaltjahrGuards`: Verwenden Sie hierfür nur Guards.
3. Funktion `schaltjahrBool`: Verwenden Sie hierfür weder $if - then - else$ -Ausdrücke noch Guards, sondern die booleschen Operatoren $\wedge$, $\vee$ und *not*.

Aufgabe 2 (lokale Deklarationen) 1 Punkt

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

   (Tutoriumsaufgabe) Zur Bestimmung der Nullstellen von quadratischen Gleichungen der Form $ax^2 + bx + c$ gibt es neben der bekannten „*p*-*q*-Formel“ (mit $p = \frac{b}{a}$ und $q = \frac{c}{a}$) auch die so genannte „*a*-*b*-*c*-Formel“

   $$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \quad .$$

   Setzen Sie diese Berechnung in Haskell zunächst in einer Funktion `abcformelLet` mit **let** um. Programmieren Sie weiterhin eine Funktion `abcformelWhere`, die die gleiche Berechnung mit **where** macht. Dabei sollen x_1 und x_2 als 2-Tupel zurückgegeben werden.

2. **Verwenden Sie für die Abgabe den Dateinamen:** `Robertson.hs`
   Bitte die Vorgabe beachten!

   Seit der Veröffentlichung der Gaußschen Wochentagsformel haben verschiedene Mathematiker und Astronomen eine ganze Reihe vergleichbarer Formeln aufgestellt. Diese benötigen meist vorausberechnete Tabellen und verwenden komplizierte Sonderregeln, z.B. für Schaltjahre. Eine im Vergleich dazu sehr elegante Formel wurde 1972 von J. D. Robertson vorgestellt. Diese wird im Folgenden beschrieben.

   Gegeben sei ein beliebiges Datum (T, M, J) im Gregorianischen Kalender, wobei T den Tag des Monats beginnend mit 1, M den Monat des Jahres beginnend mit 1 und J das Jahr bezeichnet. Dann lässt sich der Wochentag W mit $0 \triangleq$ Sonntag, $1 \triangleq$ Montag, ..., $6 \triangleq$ Samstag folgendermaßen berechnen:

$$W = \left(D + T + 77 + E + \left\lfloor \frac{B}{400} \right\rfloor - 2\left\lfloor \frac{B}{100} \right\rfloor\right) \bmod 7$$

Dabei gelte folgendes:

$$A = M + 10$$
$$B = \left\lfloor \frac{M - 14}{12} \right\rfloor + J$$
$$C = A - 12\left\lfloor \frac{A}{13} \right\rfloor$$
$$D = \left\lfloor \frac{13C - 1}{5} \right\rfloor$$
$$E = \left\lfloor \frac{5(B \bmod 100)}{4} \right\rfloor$$
$$[x] = \lfloor x \rfloor \text{ falls x} \geq 0$$
$$[x] = \lceil x \rceil \text{ falls x} < 0$$
$$\lfloor x \rfloor = max\{z \in \mathbb{Z} | z \leq x\})$$
$$\lceil x \rceil = min\{z \in \mathbb{Z} | z \geq x\}$$

Schreiben Sie eine Haskell-Funktion *robertson* :: *Integer* → *Integer* → *Integer* → *Integer*, die gemäß der obigen Formel den Wochentag berechnet, in eine Datei `Robertson.hs`.

Erweitern Sie die Klasse `Robertson.hs` um eine Methode `wochentag :: Integer -> String`, die eine Zahl zwischen 0 und 6 als Eingabe erhält und den entsprechenden Wochentag (0≙ Sonntag, 1≙ Montag, . . . , 6≙ Samstag) zurückgibt.

wochentag	(robertson	19	6	1623)	„Montag“
wochentag	(robertson	19	1	2038)	„Dienstag“
wochentag	(robertson	30	4	1777)	„Mittwoch“

Aufgabe 3 (logische Operatoren, Infix-Notation) 1 Punkt

Verwenden Sie für die Abgabe den Dateinamen: `LogOperationen.hs`
Bitte die Vorgabe beachten!

Die booleschen Funktionen *oder* sowie *nicht* seien wie folgt definiert.

```
oder :: Bool -> Bool -> Bool
oder = (||)
nicht :: Bool -> Bool
nicht = not
```

Sie können die beiden Funktionen aus der Vorgabedatei in ihre eigene Datei kopieren.

Programmieren Sie die zweistelligen booleschen Funktionen *und*, *darausFolgt*, *genauDannWenn* und *entwederOder* unter ausschließlicher Verwendung von *und* und *nicht* und ihren bereits daraus abgeleiteten Funktionen. Testen Sie Ihre Funktionen. Sie sollten im GHCi unter anderem die folgenden Testergebnisse erhalten:

$$
\begin{array}{llll}
\mathit{und} & \mathit{False} & \mathit{False} & \leadsto \mathit{False} \\
\mathit{und} & \mathit{False} & \mathit{True} & \leadsto \mathit{False} \\
\mathit{und} & \mathit{True} & \mathit{False} & \leadsto \mathit{False} \\
\mathit{und} & \mathit{True} & \mathit{True} & \leadsto \mathit{True}
\end{array}
$$

Aufgabe 4 (Rekursion über den natürlichen Zahlen) 1 Punkt

Verwenden Sie für die Abgabe den Dateinamen: `natzahlen.hs`

In den folgenden Teilaufgaben dürfen keine anderen arithmetischen Operationen als die beiden vordefinierten Funktionen *succ* und *pred* und die von Ihnen im Folgenden entworfenen Funktionen verwendet werden.

1. **(Tutoriumsaufgabe)** Programmieren Sie eine Funktion

   $$\mathit{plus} :: \mathit{Integer} \to \mathit{Integer} \to \mathit{Integer}$$

   , welche Summen natürlicher Zahlen berechnet. Nutzen Sie aus, dass $a + b = \mathit{succ}(\mathit{pred}\ a + b)$ und $0 + a = a$ gilt.

2. Programmieren Sie eine Funktion

   $$\mathit{mal} :: \mathit{Integer} \to \mathit{Integer} \to \mathit{Integer}$$

   welche Produkte natürlicher Zahlen berechnet. Nutzen Sie aus, dass $a \cdot b = b + (\mathit{pred}\ a \cdot b)$ und $0 \cdot a = 0$ bzw. $1 \cdot a = a$ gilt.

3. Programmieren Sie eine Funktion

   $$\mathit{potenz} :: \mathit{Integer} \to \mathit{Integer} \to \mathit{Integer}$$

   welche Potenzen natürlicher Zahlen berechnet. Der erste Parameter stellt die Basis dar, der zweite Parameter den Exponenten.