Institut für Informatik **Ludwig-Maximilians-Universität München**
Prof. François Bry, Thomas Prokosch
Kilian Rückschloß

Programmierung und Modellierung, SoSe 21
Übungsblatt 2

Abgabe: bis Mo 26.04.2021 08:00 Uhr

Aufgabe 2-1 Ausdrücke und Typen

Werten Sie folgende Ausdrücke aus und bestimmen Sie deren Typen (zuerst im Kopf, dann mit ghci kontrollieren!). Einen Überblick über eingebaute Typen und Funktionen in Haskell finden Sie unter anderem auf

- `https://www.haskell.org/onlinereport/haskell2010/haskellch6.html`
- `https://hackage.haskell.org/package/base-4.12.0.0/docs/Prelude.html`
- `https://hoogle.haskell.org/?scope=package%3Abase`

a) `compare 2 3`

b) `13 `mod` 4`

c) `(-) 7 4`

d) `odd 3`

e) `even 3`

f) `drop 3 [1, 2, 3, 4, 5]`

g) `elem 3 [1, 2, 3, 4, 5]`

h) `sum [1, 2, 3, 4, 5]`

i) `reverse "Lag er im Kajak, mir egal"`

j) `words "Lag er im Kajak, mir egal"`

k) `unlines ["Lag", "er", "im", "Kajak,", "mir", "egal"]`

Aufgabe 2-2 Funktionen

a) Schreiben Sie eine Funktion `alleGleich :: Eq a => a -> a -> a -> Bool`, die überprüft, ob drei Integer-Zahlen gleich sind.

b) Schreiben Sie eine Funktion `ungerade :: Integral a => a -> Bool`, die überprüft, ob eine Integer Zahl ungerade ist. Verwenden Sie hierzu nicht die vordefinierten Funktionen `even` oder `odd`.

c) Schreiben Sie eine Funktion `gerade :: Integral a => a -> Bool`, die überprüft, ob eine Integer Zahl gerade ist. Verwenden Sie hierzu nicht die vordefinierten Funktionen `odd` oder `even`.

Aufgabe 2-3 List-Comprehension

In den folgenden Unteraufgaben sollen Sie Funktionen mit der Hilfe von List-Comprehensions definieren. Dabei dürfen nur die selbstgeschriebenen Funktionen aus Aufgabe 2-2 und die folgenden vordefinierten Funktionen verwendet werden: `+, -, *, /, ==, /=, sum, mod, elem, putStrLn, maximum` und die Funktionen aus Aufgabe 2-2 verwendet werden.

Anmerkung: Ziel dieser Aufgabe ist das Einüben von List-Comprehensions, nicht das Erstellen möglichst effizienter Programme.

a) Definieren Sie eine Funktion, die alle ungeraden Elemente einer Liste (von Zahlen) ausgibt.

b) Definieren Sie eine Funktion, die zu einer Liste (von Zahlen) alle ihrer geraden Zahlen und das Zweifache ihrer ungeraden Zahlen ausgibt. Beispiel: `geradeZahlen2 [1,4,7,8]` liefert die Liste `[2,4,14,8]`

c) Definieren Sie eine Funktion, die für alle natürlichen Zahlen im Bereich zwischen zwei Zahlen (einschließlich dieser) eine Liste jener Zahlen zurückgibt, bei denen die Division durch 7 Rest 5 ergibt. Es darf vorausgesetzt werden, dass der erste Parameter immer kleiner als der zweite ist.

   Beispiel: `div7Rest5 7 28` ergibt `[12,19,26]`

d) Definieren Sie eine Funktion, welche die Länge einer Liste berechnet, ohne die vordefinierte Funktion `length` zu benutzen.

e) Definieren Sie ein Funktion 'dreifach', die einen Integer Wert als Eingabe erhält und dessen dreifachen Wert zurück gibt (siehe Übungsblatt 1 Aufgabe 2a).

f) Definieren Sie ein Funktion `nurGrossBuchstaben :: [Char] -> [Char]`, die nur die Großbuchstaben eines Strings ausgibt.

   Definieren Sie eine zweite Funktion, `putStrLnNurGrossBuchstaben :: [Char] -> IO ()`, die die Funktion `nurGrossBuchstaben` und deren Ergebnis auf die Konsole ausgibt. Benutzen Sie hierfür den "." Operator.

g) Definieren Sie eine Funktion, welche die Faktorzerlegung einer natürlichen Zahl als Liste zurückgibt.
   Beispiel: `faktoren 20` liefert die Liste `[2,4,5,10]`.

h) Definieren Sie eine Funktion, welche den größten gemeinsamen Teiler zweier natürlicher Zahlen als Liste zurückgibt. Benutzen Sie die Funktion faktoren aus der vorherigen Aufgabe.

i) Pythagoreische Tripel: Definieren Sie ein Funktion, welche einen Integer Wert n als Eingabe erhält und eine Liste mit allen Trippeln *(a, b, c)* mit $a, b, c \leq n$ zurück gibt, die den Satz des Pythagoras erfüllen: $a^2 + b^2 = c^2$
   Beispiel: `pytri 10` liefert eine Liste mit den Elementen `(3,4,5) und (6,8,10)` zurück.

Aufgabe 2-4 Ein- und Ausgabe: Palindrom

a) Im Verzeichnis `u02/2-4/` befindet sich eine Datei `palindrom.txt` mit einigen Wörtern, wobei in jeder Zeile genau ein Wort steht.
Schreiben Sie ein Programm, das alle Wörter in der Datei auf der Standardausgabe ausgibt und speichern Sie das Programm in der Datei `u02/2-4/palindrom-a.hs` ab.

b) Erweitern Sie das Programm, dass es zu den Wörtern immer auch deren Länge ausgibt und speichern Sie das Programm in der Datei `u02/2-4/palindrom-b.hs` ab.

Beispielausgabe: `regallager hat die Laenge 10`

c) In der Vorlesung haben Sie das Programm `palydrom.hs` kennengelernt, das Worte von der Standardeingabe einliest und testet, ob diese Palindrome sind.

Verändern Sie das Programm so, dass es die Wörter aus der Datei `palindrom.txt` einliest und ausgibt, ob es sich um Palindrome handelt. Zusätzlich soll auch die Länge der Wörter berechnet und ausgegeben werden.

Speichern Sie das Programm in der Datei `u02/2-4/palindrom-c.hs` ab.

Beispielausgabe: `regallager ist ein Palindrom und hat die Laenge 10`

Aufgabe 2-5 Ein- und Ausgabe: Übersetzer

a) Schreiben Sie ein Programm, das erst ein „hochdeutsches" Wort von der Standardeingabe einliest und dann die „bairische" Übersetzung erfragt. Das Programm soll so lange nach Wörtern fragen, bis der Nutzer keine mehr eingibt. Das Programm soll nach Eingabe eines Wortpaares zur Bestätigung eine Nachricht im Format *"<Hochdeutsches Wort> heißt auf Bairisch <bairische Übersetzung>"* auf die Konsole ausgeben.

Speichern Sie das Programm in der Datei `u02/2-5/woerterbuch-a.hs` ab. Beispiel Ablauf:

```
Deutsches Wort:
hallo
Bairisches Wort:
servus
'hallo' heisst auf Bairisch 'servus'
Deutsches Wort:
```

b) Erweitern Sie das Programm, sodass es die hochdeutsch-bairischen Wort-Tupel zeilenweise in eine Datei `woerterbuch.txt` speichert.

Speichern Sie das Programm in der Datei `u02/2-5/woerterbuch-b.hs` ab.

Inhalt der `woerterbuch.txt` Datei aus dem Verzeichnis `u02/2-5/` :

```
hallo servus
gut guad
vater voda
```