WS 2020/2021

Funktionale Programmierung

Lösungen zum **0. Übungsblatt**

Prof. Dr. Margarita Esponda

Ziel: Auseinandersetzung mit der Haskell-Syntax, vordefinierten Haskell-Funktionen und ersten einfachen Funktionsdefinitionen.

1. Aufgabe

Haskell installieren (aus http://www.haskell.org).

2. Aufgabe

Verwenden Sie Haskell als Taschenrechner und berechnen Sie folgende Ausdrücke. Erläutern Sie kurz die Ergebnisse oder die Fehlern, die dabei angezeigt werden.

Lösungen für einige Teilaufgaben:

Ausdruck	Wert	Bagründung
2**1024	Infinity	Das Ergebnis des Ausdrucks überschreitet die größte Gleitkomma-Zahl, die in Haskell mit 64 Bits kodiert werden kann, und deswegen wird nach dem IEEE-Standard Infinity als Wert zurückgegeben.
div 5 (-2)	-3	Die **div**-Funktion macht eine ganzzahlige Division und rundet negative Zahlen immer in Richtung **-Infinity**.
0.9==0.3*3	False	Die Dezimalzahlen können nicht immer exakt in Binärzahlen umgewandelt werden. Die Rundungsfehler, die dabei entstehen, verursachen, das Ausdrücke, die eigentlich dem gleichen Wert entsprechen, am Ende unterschiedlich sind. Gleitkomma-Zahlen sollten deswegen nicht nach Gleichheit verglichen werden. Z.B. **0.1_{10}** ergibt **$0.000110011001\underline{1100}\ldots_2$** (periodische Binärzahl)
abs -7	**Fehler!**	Die **Funktionsapplikation in Haskell ist linksassoziativ**. Der Interpreter versucht zuerst den Ausdruck (abs -) zu berechnen, aber die abs-Funktion ist nicht für den Operator (-) definiert. Der richtige Ausdruck muss "abs (-7)" heißen.
mod 5 (-2)	-1	Die **mod**-Funktion ist so definiert, dass, wenn y ungleich 0 ist, folgende Gleichung erfüllt werden sollte: **(x `div` y)*y + (x `mod` y) == x** Die **div**-Funktion macht eine ganzzahlige Division und rundet negative Zahlen immer in Richtung -Infinity.
rem 5 (-2)	1	Die **rem**-Funktion ist so definiert, dass, wenn y ungleich 0 ist, folgende Gleichung erfüllt werden sollte: **(x `quot` y)*y + (x `rem` y) == x** Die **quot**-Funktion macht eine ganzzahlige Division und rundet negative Zahlen immer in Richtung **0**.
-3 `mod` 5	-3	Die Priorität der **mod**-Funktion ist höher als die Priorität der (-) Funktion. D.h. **-3 `mod` 5** equiv. **-(3 `mod` 5)**
sqrt (-1)	NaN	Die Quadratwurzel einer negativen Zahl ist für die realen Zahlen nicht definiert.
exp 1	2.71821...	Exponential-Funktion, die angewendet auf 1 der Euler-Zahl entspricht.

3. Aufgabe

Was ist der **Wert** folgender Ausdrücke? Versuchen Sie, zuerst die Lösungen mit Zwischenschritte zu schreiben, ohne in dem Haskell-Interpreter die Ausdrücke einzugeben. Oder begründen Sie Ihre Antworten.

```
(-) ((+) ((+) 1 2) 3) (-2)
(-4 `mod` 5) == (-4 `rem` 5)
(4 `mod` (-5)) == (4 `rem` (-5))
4 == (div 4 (-3))*(-3) + (mod 4 (-3))
succ 4 * 8 == succ (4 * 8)
(10**17)*((0.1)*3-(0.1)*2-(0.1))
log 0
```

Lösungen:

```
(-) ((+) ((+) 1 2) 3) (-2)  => (-) ((+) 3 3) (-2)
                            => (-) 6 (-2)
                            => 8

(-4 `mod` 5) == (-4 `rem` 5) => -(4 `mod` 5) == -(4 `rem` 5)
                             => - 4 == - 4
                             => True

(4 `mod` (-5)) == (4 `rem` (-5)) => -1 == 4
                                 => False

succ 4 * 8 == succ (4 * 8)  =>  (succ 4) * 8 == succ (4 * 8)
                            => 5 * 8 == succ 32
                            => 40 == 33
                            => False

4 == (div 4 (-3))*(-3) + (mod 4 (-3)) => 4 == (-2)*(-3) + (-2)
                                      => 4 == 6 - 2
                                      => 4 == 4
                                      =>  True

(10**17)*((0.1)*3-(0.1)*2-(0.1))  =>  (10**17)*(0.3 - 0.2 - 0.1)
                                  =>  (10**17)*(-2.7755575615628914e-17)
                                  => -2.7755575615628914 (Haskell-Berechnung)
```

log 0 nach der mathematische Definition der Logarithmus-Funktion gibt es keine Lösung für die Gleichung $\mathbf{e^x = 0}$. Das bedeutet **log 0** ist nicht definiert. Haskell reduziert aus praktischen Gründen (IEEE) **log 0** zu **-Infinity**.

4. Aufgabe

Warum ist **(min -2 0)** kein gültiger Haskell-Ausdruck in Prelude?

Lösung:

Die **Funktionsapplikation ist linksassoziativ**. Der Interpreter versucht zuerst den Ausdruck (min -) zu berechnen, aber die min-Funktion ist nicht für den (-) Operator definiert. Der richtige Ausdruck würde "**min (-2) 0**" heißen.

b) Warum ist der Ausdruck **(mod 1 0)** fehlerhaft?

Lösung:

Weil die **div x 0** nicht definiert ist. Z.B. div 3 0 => *** Exception: divide by zero.

Warum ist **(0.1 == 0.3/3)** oder **0.9 == 3*(0.3)** gleich **False**?

Lösung:

Die Dezimalzahlen können nicht immer exakt in Binärzahlen umgewandelt werden. Die Rundungsfehler, die dabei entstehen, verursachen, das Ausdrücke, die eigentlich dem gleichen Wert entsprechen, am Ende unterschiedlich sind. Gleitkomma-Zahlen sollten deswegen nicht nach Gleichheit verglichen werden. Z.B.

0.1_{10} ergibt **$0.000110011001\underline{1100}\ldots_2$** (periodische Binärzahl)

Warum sind die Ausdrücke **quot 1.0 3** und **3^1.0** fehlerhaft?

Lösung:

Weil beide Ausdrücke Datentyp-Fehler beinhalten. Die **quot** Funktion ist nur für ganze Zahlen als Argumente definiert und die **^**(Potenzfunktion) erlaubt nur ganze Zahlen als Potenz.

5. Aufgabe

Der Body-Mass-Index eine Person wird nach folgende Formel berechnet:

BMI = Körpergewicht in Kg./ (Körpergrößen in m.)^2

Definieren Sie eine Funktion **body_mass_index** in Haskell, die bei Eingabe des Körpergewichts und der Körpergröße einer Person den Body-Mass-Index berechnet.

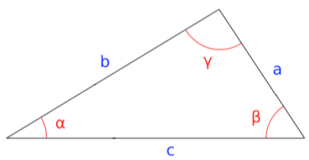
Lösung:

```
body_mass_index :: Double -> Double -> Double
body_mass_index weight height = weight / (height^2)
```

6. Aufgabe (5 Punkte)

Der Flächeninhalt eines Dreiecks kann mit Hilfe der Heron Formel wie folgt berechnet werden:

$$Fläche_{\triangle} = \sqrt{s(s-a)(s-b)(s-c)} \quad \text{mit} \quad s = \frac{a+b+c}{2}$$



Lösung:

```
triangle_area :: Double -> Double -> Double -> Double
triangle_area a b c = sqrt (s*(s-a)*(s-b)*(s-c))
                      where
                      s = (a + b + c) / 2.0
```

7. Aufgabe (5 Punkte)

Definieren Sie eine Haskell-Funktion, die die Windchill-Temperatur (WCT) mit Hilfe folgenden Formel berechnet:

$$WCT = 13{,}12 + 0{,}6215 \cdot T - 11{,}37 \cdot v^{0{,}16} + 0{,}3965 \cdot T \cdot v^{0{,}16}$$

mit T = Lufttemperatur in Grad-Celsium

v = Windgeschwindigkeit in Kilometer pro Stunde

Lösung:

```
wct :: Double -> Double -> Double
wct temperatur velocity = a + b*temperatur + (e*temperatur-c)*velocity**d
                          where
                           a = 13.12
                           b = 0.6215
                           c = 11.37
                           d = 0.16
                           e = 0.3965
```

Wichtige Hinweise:

1) Verwenden Sie geeignete Namen für Ihre Variablen und Funktionsnamen, die den semantischen Inhalt der Variablen oder die Semantik der Funktionen wiedergeben.

2) Verwenden Sie vorgegebene Funktionsnamen, falls diese angegeben werden.

3) Kommentieren Sie Ihre Programme.

4) Verwenden Sie geeignete lokale Funktionen und Hilfsfunktionen in Ihren Funktionsdefinitionen.

5) Schreiben Sie in alle Funktionen die entsprechende Signatur.