INFORMATIK I

Tutorium 5 — 15. November 2024

Aufgepasst!

Name Last
Universität Münster

LaTeX-Vorlage von
Florian Sihler

Übungsblatt 4

1 ○

Aufgabe 1

Setzen Sie Klammern, um zu zeigen, in welcher Reihenfolge die Teilausdrücke ausgewertet werden, und geben Sie das Ergebnis der Auswertung an.

Lösung 1: a) `3 * 4 - 2 ^ 2 ^ 4`

- **Potenz**: Bindungsstärke 8, rechts-assoziativ
- **Multiplikation**: Bindungsstärke 7
- **Subtraktion**: Bindungsstärke 6

```
3 * 4 - (2 ^ (2 ^ 4))
(3 * 4) - (2 ^ (2 ^ 4))
(3 * 4) - (2 ^ (2 ^ 4))
```

$= -65524$

Aufgabe 1

Setzen Sie Klammern, um zu zeigen, in welcher Reihenfolge die Teilausdrücke ausgewertet werden, und geben Sie das Ergebnis der Auswertung an.

Lösung 1: b) `add 5 3 + add 7 4 * 2 - 3`

- **Funktion**: Bindungsstärke 10 `(add 5 3) + (add 7 4) * 2 - 3`
- **Multiplikation**: Bindungsstärke 7 `(add 5 3) + ((add 7 4) * 2) - 3`
- **Addition/Subtraktion**: Bindungsstärke 6, daher Assoziativität (hier: links)

`((add 5 3) + ((add 7 4) * 2)) - 3`

$= 27$

Aufgabe 2: a) + b)

Programmieren Sie in Haskell einen zweistelligen, links-assoziativen Infix-Operator ~~ mit der Bindungsstärke 5, der den Mittelwert zweier Float-Zahlen berechnet.

Lösung 2: a) + b)

```haskell
-- Bindungsstärke 5
infixl 5 ~~

-- Signatur, Operator in Klammern
(~~) :: Float -> Float -> Float

-- Mittelwert: Summe, geteilt durch 2
x ~~ y = (x + y)/2
```

Aufgabe 2: c)

Geben Sie an, wie der Ausdruck `3 + 4 ~~ 5 * 6` ausgewertet wird. Welche Änderungen würden sich bei den Bindungsstärken 6 oder 7 ergeben?

Lösung 2: c)

- Bindungsstärke 5: $* > + > \sim\sim$: $(3 + 4) \sim\sim (5 * 6) = 18.5$
- Bindungsstärke 6: $* > +|\sim\sim$, also links-assoziativ:
  $(3 + 4) \sim\sim (5 * 6) = 18.5$
- Bindungsstärke 7: $*|\sim\sim > +$, also links-assoziativ:
  $3 + ((4 \sim\sim 5) * 6) = 30$

Aufgabe 3

Betrachten Sie eine zweistellige Funktion `power`, die für natürliche Zahlen b und e den Wert b^e berechnet.

Lösung 3: a) Geben Sie `power` als mathematische Funktion an

$$power : \mathbb{N} \times \mathbb{N} \to \mathbb{N},\ (b, e) \mapsto b^e$$

Aufgabe 3

Betrachten Sie eine zweistellige Funktion `power`, die für natürliche Zahlen b und e den Wert b^e berechnet.

Lösung 3: b) Programmieren Sie `power` in haskell

```
power :: Integer -> Integer -> Integer
power b e =
    if e >= 0 && b >= 0
    then b ^ e
    else -1
```

- Wegen *Currying* gibt es nur einstellige Funktionen: $\mathbb{D} =$ `Integer`.
- Zielmenge: Funktion $\mathbb{W} = \{$`Integer -> Integer`$\}$

Aufgabe 3

Programmieren Sie die einstellige Funktion `powerOfTwo`, die für eine natürliche Zahl e den Wert 2^e berechnet.

Lösung 3: c)

```
-- mit Parameter
powerOfTwoA :: Integer -> Integer
powerOfTwoA e = power 2 e

-- wir nutzen Currying
powerOfTwoB :: Integer -> Integer
powerOfTwoB = power 2
```

Aufgabe 4: a)

Die Veränderung der verkauften Portionen bei einer Preiserhöhung um x wird besser durch die Funktion $f(x) = x^3 + ax$ beschrieben.

Lösung 4: a) Berechnen Sie a, so dass $f(1) = 10$.

$$f(x) = x^3 + ax$$

$$f(1) = 1^3 + a \cdot 1$$

$$1 + a = 10 \implies a = 9$$

Aufgabe 4: b)

Es können maximal 300 Gerichte verkauft werden, mit einem Mengenrabatt von 5% ab 150 Gerichten.

Lösung 4: b) Ergänzen Sie das Programm um geeignete Konstanten

```haskell
-- Mengenrabatt
discount :: Double
discount = 0.05

-- Schwellwert
discount_threshold :: Int
discount_threshold = 150

-- Maximalmenge
max_portions :: Int
max_portions = 300
```

Aufgabe 4: c)

Passen Sie das Programm an und führen Sie neue Funktionen mit passenden Beispielen ein.

Lösung 4: c) Wir haben bereits...

```haskell
-- Mengenrabatt
discount :: Double
discount = 0.05

-- Schwellwert
discount_threshold :: Int
discount_threshold = 150

-- Maximalmenge
max_portions :: Int
max_portions = 300
```

Aufgabe 4: c)

Passen Sie das Programm an und führen Sie neue Funktionen mit passenden Beispielen ein.

Lösung 4: c) Einfach übernehmen:

```haskell
-- Einkaufspreis
price_per_portion :: Double
price_per_portion = 2.0

-- Grundkosten
base_costs :: Double
base_costs = 200.0

-- aktueller Verkaufspreis
base_price :: Double
base_price = 9.90

-- Grundanzahl Portionen
base_portions :: Int
base_portions = 100
```

Aufgabe 4: c)

Passen Sie das Programm an und führen Sie neue Funktionen mit passenden Beispielen ein.

Lösung 4: c) Die neue Formel:

```haskell
-- Verkaufsrückgang pro Euro Preisanstieg
demand_decrease :: Double -> Int
demand_decrease delta = round (delta^3 + 9*delta)

{-
-- wir entfernen dafür dec_portions
dec_portions :: Int
dec_portions = 10
-}
```

Aufgabe 4: c)

Passen Sie das Programm an und führen Sie neue Funktionen mit passenden Beispielen ein.

Lösung 4: c) Obergrenze für den Verkauf:

```haskell
-- Beschränkt die Anzahl der Portionen
capped_portions :: Int -> Int
capped_portions portions
    | portions < 0             = 0
    | portions > max_portions = max_portions
    | otherwise                = portions
```

Aufgabe 4: c)

Passen Sie das Programm an und führen Sie neue Funktionen mit passenden Beispielen ein.

Lösung 4: c) Rabatt einberechnen:

```haskell
-- Berechnet Preis inklusive Rabatt
portion_price :: Int -> Double
portion_price meals =
    if meals >= discount_threshold
    then price_per_portion * (1 - discount)
    else price_per_portion
```

Aufgabe 4: c)

Passen Sie das Programm an und führen Sie neue Funktionen mit passenden Beispielen ein.

Lösung 4: c) `portions` und `costs`:

```
-- Verkaufte Portionen für einen Preis
portions :: Double -> Int
portions price = capped_portions (base_portions - ↩
   demand_decrease (price - base_price))

-- Ausgaben für einen Preis
costs :: Double -> Double
costs price = base_costs + fromIntegral (portions price) * ↩
   portion_price (portions price)
```

Aufgabe 4: c)

Passen Sie das Programm an und führen Sie neue Funktionen mit passenden Beispielen ein.

Lösung 4: c) Der Rest:

```
-- Einnahmen für einen Preis
revenue :: Double -> Double
revenue price = price * fromIntegral (portions price)

-- Gewinn für einen Preis
profit :: Double -> Double
profit price = (revenue price) - (costs price)
```

Anhang

ANHANG

- Es gibt eine LaTeX-Vorlage für Übungsblätter von der AG Vahrenhold:

(hier die `minted`-Version)

Interesse an einer LaTeX-Einführung? :D

Name Last
Münster, 17. November 2024
name.last@uni-muenster.de