

# Informatik I - Gruppe 1 - Übungsblatt 6

Ausgabe: 14.11.2022

Abgabe: 21.11.1022

Tutor: Tim Völker

Ali Kurt 528961

Thomas Kujawa 463620

Felix Hoff 366927

## Aufgabe T6.1: Rekursion (3 Punkte)

Gegeben sei eine Funktion  $f$ , definiert durch:

$$f(n) = \begin{cases} 2 \cdot f(n-1) + 1 & \text{für } n \geq 1 \\ 1 & \text{sonst} \end{cases}$$

- (a) Geben Sie eine Wertetabelle an, die für  $n = 0, 1, 2, 3, 4, 5, 6$  das Ergebnis  $f(n)$  angibt.
- (b) Geben Sie den schrittweisen Ablauf der rekursiven Funktion als Expansion und Kontraktion für den Wert  $n = 4$  an.
- (c) In welchem Verhältnis oder mathematischen Zusammenhang stehen  $n \geq 0$  und der Funktionswert  $f(n)$  ? Geben Sie die Funktion  $f(n)$  in einer nicht-rekursiven Form an.

# Informatik I - Gruppe 1 - Übungsblatt 6

Ausgabe: 14.11.2022

Abgabe: 21.11.1022

Tutor: Tim Völker

Ali Kurt 528961

Thomas Kujawa 463620

Felix Hoff 366927

---

## Aufgabe T6.2: ADT Spezifikation (4 Punkte)

In der Vorlesung haben Sie abstrakte Datentypen (ADT) kennengelernt. Dabei wurde in der Vorlesung die Spezifikation eines Datentyps  $A$  vorgestellt (Folie 291). Sie sollen nun 2 Realisierungen des Datentyps  $A$  angeben, die nicht bereits in der Vorlesung vorgestellt wurden (Bool, Nat0 und anderen Datentypen aus Num) und welche die Signatur des Datentyps  $A$  realisieren. Überprüfen Sie für beide Realisierungen, ob diese die Spezifikation erfüllen.

# Informatik I - Gruppe 1 - Übungsblatt 6

Ausgabe: 14.11.2022

Abgabe: 21.11.1022

Tutor: Tim Völker

Ali Kurt 528961

Thomas Kujawa 463620

Felix Hoff 366927

## Aufgabe P6.3: Binomialkoeffizienten / Pascalsches Dreieck (2 Punkte)

Zur Ermittlung der Binomialkoeffizienten  $\binom{n}{k}$  kann das Pascalsche Dreieck zu Hilfe gezogen werden. Den passenden Koeffizienten findet man in der  $n$ -ten Zeile an der  $k$ -ten Stelle:

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Dabei kann dies mathematisch durch folgende Funktion  $f$  definiert werden:

$$\begin{aligned} f(n, 0) &= 1 \\ f(n, n) &= 1 \\ f(n, k) &= \begin{cases} 0 & \text{für } k > n \\ f(n-1, k-1) + f(n-1, k) & \text{sonst} \end{cases} \end{aligned}$$

Implementieren Sie die Funktion `pascal::Int → Int → Int`, die die Parameter  $n$  und  $k$  entgegennimmt und den Binomialkoeffizienten zurückliefert. Implementieren Sie dies als rekursive Funktion. Geben Sie die Berechnung für `pascal (3, 1)` als Kommentar an. Geben Sie außerdem an, ob es sich um eine direkte oder indirekte Rekursion handelt.

```
1 pascal :: Int -> Int -> Int
2 pascal n k
3   | k == 0 = 1
4   | n == k = 1
5   | otherwise = if k>n
6                 then 0
7                 else
8                   pascal (n-1) (k-1) + pascal (n-1) k
```

## Aufgabe P6.4: Rationale Zahlen (3 + 3 = 6 Punkte )

In der Vorlesung haben Sie eine Haskell-Implementierung der rationalen Zahlen kennengelernt. In dieser Aufgaben soll die Implementierung des Datentyps `Ratio` erweitert werden. Folgen Sie dabei dem Programm-entwurf der Vorlesung und erstellen Sie (1) Funktionsköpfe, (2) Beispiele und (3) Funktionsrümpfe sowie (4) eine Überprüfung anhand der generierten Beispiele.

(a) Entwickeln Sie eine Funktion `reduceFraction`, die eine rationale Zahl entgegen nimmt und die vollständig gekürzte rationale Zahl zurück gibt. Implementieren Sie dazu die Hilfsfunktion `ggT`, die den größten gemeinsamen Teiler zweier Zahlen  $a$  und  $b$  bestimmt, indem sie  $a$  zurück gibt, falls  $b = 0$ . Ansonsten soll der größte gemeinsame Teiler von  $b$  und  $a \% b$  bestimmt werden.

(b) Entwickeln Sie Funktionen `fromRatio` und `toRatio`, die eine Konvertierung zwischen dem Datentyp `Float` und den rationalen Zahlen vornehmen. Die rationale Zahl soll dazu in ihrer vollständig gekürzten Form zurück gegeben werden. Nicht jede rationale Zahl kann als Gleitkommazahl exakt dargestellt werden. Runden Sie die Gleitkommazahlen dazu im Kontext dieser Aufgabe auf 5 Nachkommastellen.

Hinweis: Sie können die Haskell-Datei aus dem Learnweb als Startpunkt nutzen.

```
1 import Distribution.Simple.Flag (Flag)
2 -- Datentyp für rationale Zahlen repräsentiert durch zwei Ganzzahlen für Zähler und Nenner
3 data Ratio = MkRatio Integer Integer
4
5
6 -- Gleichheit für rationale Zahlen
7 instance Eq Ratio where
8     (MkRatio nom1 den1) == (MkRatio nom2 den2) = (nom1 * den2 == nom2 * den1)
9
10 -- Ordnung für rationale Zahlen
11 instance Ord Ratio where
12     (MkRatio nom1 den1) <= (MkRatio nom2 den2) =
13         if ((signum den1) * (signum den2) == 1)
14             then (nom1 * den2 <= nom2 * den1)
15             else (nom2 * den1 <= nom1 * den2)
16
17 -- Rechenoperationen für rationale Zahlen inklusive Addition, Multiplikation, Negierung,
18   ↳ Betragsfunktion, Vorzeichenfunktion und Konvertierung von Integer
19 instance Num Ratio where
20     (+) (MkRatio nom1 den1) (MkRatio nom2 den2) = (MkRatio (nom1 * den2 + nom2 * den1) (den1 *
21   ↳ den2))
22     (*) (MkRatio nom1 den1) (MkRatio nom2 den2) = (MkRatio (nom1 * nom2) (den1 * den2))
23     negate (MkRatio nom den) = (MkRatio (- nom) den)
24     abs (MkRatio nom den) = (MkRatio (abs nom) (abs den))
25     signum (MkRatio nom den) = (MkRatio ((signum nom) * (signum den)) 1)
26     fromInteger i = (MkRatio i 1)
```

# Informatik I - Gruppe 1 - Übungsblatt 6

Ausgabe: 14.11.2022

Abgabe: 21.11.1022

Tutor: Tim Völker

Ali Kurt 528961

Thomas Kujawa 463620

Felix Hoff 366927

---

```
25
26
27
28 -- Funktion zum Anzeigen von rationalen Zahlen
29 -- Die Notation wird später in der Vorlesung erläutert
30 instance Show Ratio where
31     show (MkRatio nom den) = (show nom) ++ "/" ++ (show den)
32
33 ggT :: Integer -> Integer -> Integer
34 ggT a b = if b == 0 then a
35           else ggT b (mod a b)
36
37 myRatio :: Ratio
38 myRatio = MkRatio 2 3
39
40 reduceFraction :: Ratio -> Ratio
41 reduceFraction (MkRatio nom den) = MkRatio (div nom (ggT nom den)) (div den (ggT nom den))
42
43 fromRatio :: Ratio -> Float
44 fromRatio (MkRatio nom den) = roundFive (fromIntegral nom / fromIntegral den)
45
46 roundFive :: Float -> Float
47 roundFive f = fromIntegral(round (f*10^5))/(10^5)
48
49 decimalPlace :: Float -> Integer
50 decimalPlace f = truncate (roundFive f*10^5)
51
52 toRatio :: Float -> Ratio
53 toRatio f = reduceFraction (MkRatio (decimalPlace f) 100000)
54
```

---

**Aufgabe P6.5:** Rekursive Datentypen ( $1 + 3 + 1, 5 + 1, 5 + 3 = 10$  Punkte )

In der Vorlesung haben Sie den ADT Binärbaum Btree kennengelernt. Sie sollen nun einen konkreten Datentypen BtreeFloat implementieren, der nur Elemente vom Typ Float abspeichert. Folgen Sie dabei dem Programmmentwurf der Vorlesung und erstellen Sie (1) Funktionsköpfe, (2) Beispiele und (3) Funktionsrümpfe sowie (4) eine Überprüfung anhand der generierten Beispiele. Bearbeiten Sie dafür folgende Aufgaben:

- (a) Implementieren Sie nun den Binärbaum BtreeFloat. Ein Binärbaum ist entweder ein leerer Knoten Nil oder ein Knoten Node mit einer Gleitkommazahl (Float) als Wert und einem linken Kind (einem Binärbaum) und einem rechten Kind (einem Binärbaum).
- (b) Implementieren Sie die Funktion  $\text{insert} :: \text{BtreeFloat} \rightarrow \text{Float} \rightarrow \text{BtreeFloat}$  für den Datentyp BtreeFloat. Diese Funktion fügt eine Gleitkommazahl in den Binärbaum ein. Für jeden Knoten wird der Wert  $s$  mit dem Wert des aktuellen Knoten  $s_i$  verglichen. Ist  $s < s_i$  dann wird der Wert in den linken Binärbaum eingefügt, ist  $s > s_i$  dann wird der Wert in den rechten Binärbaum eingefügt. Sind beide Werte gleich wird der Wert nicht eingefügt. Der Wert  $s$  wird soweit nach unten durchgereicht bis ein Blatt erreicht wird. Dort wird dann ein neuer Knoten mit dem Wert  $s$  und zwei leeren Kindern (Binärbäumen) eingefügt.
- (c) Implementieren Sie die Funktion  $\text{depth} :: \text{BtreeFloat} \rightarrow \text{Int}$  für den Datentypen BtreeFloat. Diese Funktion gibt die Tiefe eines Binärbaumes an. Dabei ist die Tiefe beschrieben durch den maximal längsten Pfad von der Wurzel bis zu einem Blatt.
- (d) Implementieren Sie die Funktion  $\text{max\_node} :: \text{BtreeFloat} \rightarrow \text{Float}$  für den Datentypen BtreeFloat. Diese Funktion gibt den größten Knotenwert innerhalb des Binärbaumes zurück. Sollte die Eingabe nur ein leerer Knoten sein, dann soll max\_node den Wert 0 zurückgeben. Hinweis: Sie können davon ausgehen, dass der Binärbaum mit der Funktion insert erstellt wurde.
- (e) Implementieren Sie die Funktion  $\text{path\_diff} :: \text{BtreeFloat} \rightarrow \text{Int}$  für den Datentypen BtreeFloat. Diese Funktion gibt die Differenz zwischen dem kürzesten und dem längsten Pfad im Binärbaum zurück. Ein Pfad sei dabei die Anzahl der Knoten vom Wurzelknoten (dem Startknoten) zu einem Blattknoten.

