WS 2020/2021

Funktionale Programmierung

5. Übungsblatt

Prof. Dr. Margarita Esponda

Ziel: Arbeiten mit Funktionen höherer Ordnung, endrekursiven Funktionen und Komplexitätsanalyse von Funktionen.

1. Aufgabe (3 Punkte)

Im Haskell-Prelude ist die **iterate**-Funktion wie folgt definiert:

```
iterate      :: (a -> a) -> a -> [a]
iterate f x  =  x : iterate f ( f x )
```

Definieren Sie unter Verwendung der **iterate**-Funktion Funktionen, die folgende unendlichen Listen darstellen:

[1, -1, 1, -1, 1, -1, …]
[0, 1, 3, 7, 15, 31, 63, …]
[(0,1), (1,1), (1,2), (2,3), (3,5), (5,8), …]

2. Aufgabe (4 Punkte)

Analysieren Sie die Komplexität folgender zwei Multiplikationsfunktionen:

```
mult :: Integer -> Integer -> Integer
mult n 0 = 0
mult n m = mult n (m-1) + n

russMult :: Integer -> Integer -> Integer
russMult n 0 = 0
russMult n m | (mod m 2)==0  = russMult (n+n) (div m 2)
             | otherwise     = russMult (n+n) (div m 2) + n
```

3. Aufgabe (6 Punkte)

a) Schreiben Sie eine Haskell Funktion, das einen Text als Eingabe bekommt und alle Worte, die sich mit den letzen drei Buchstaben reimen, in eine Liste von Gruppenworten klassifiziert.

Anwendungsbeispiel:

classifyRhymeWords "Nikolaus baut ein Haus aus Holz und klaut dabei ein Bauhaus." =>

[[“klaut","baut"],["Nikolaus","Bauhaus","Haus","aus"],["ein","ein"],["dabei"],["und"],["Holz"]]

b) Analysieren Sie die Komplexität der Funktion.

4. Aufgabe (6 Punkte)

Der Selectionsort Algorithmus sucht das kleinste/größte Element in der zu sortierenden Liste, platziert dieses am Anfang der Liste und wiederholt das Verfahren mit dem Rest der Liste.

a) Definieren Sie eine polymorphe Funktion, die unter Verwendung des Selectionsort-Algorithmus die Elemente einer gegebenen Liste sortiert. Ein zweites Argument, das eine

Vergleichsoperation sein soll, entscheidet, ob die Liste in absteigender oder ansteigender Reihenfolge sortiert wird. Verwenden Sie in Ihrer Definition eine lokale Hilfsfunktion **calculateFirst**, die je nach angegebener Vergleichsoperation das kleinste oder das größte Element der Liste findet, und eine zweite lokale Funktion **deleteElem**, die das gefundene Element aus der Restliste entfernt.

Anwendungsbeispiele:

selectionSort (<) [2,1,5,0,4,3,7] => [0,1,2,3,4,5,7]
selectionSort (>) [2,1,0,4,3,7] => [7,5,4,3,2,1,0]

b) Analysieren Sie die Komplexität der Funktion.

5. Aufgabe (3 Punkte + 3 Bonuspunkte)

a) (3 Punkte) Definieren Sie eine Funktion **pyth_tripels** die bei Eingabe einer natürlichen Zahl **n**, die Liste aller (a, b, c) Pythagoras-Zahlentripel mit $0 < a \leq b \leq c \leq n$ ohne Wiederholungen berechnet.

Anwendungsbeispiel:

pyth_tripels 16 => [(3,4,5),(5,12,13),(6,8,10),(9,12,15)]

b) (3 Bonuspunkte) Analysieren Sie die Komplexität der **pyth_tripels** Funktion.

6. Aufgabe (3 Punkte)

In der Bildverarbeitung werden für die Darstellung von Farben unterschiedliche Formate verwendet.

Es gibt z.B. das RGB-Format, in dem mit Hilfe von drei ganzen Zahlen zwischen 0 und 255 (Rot, Grün und Blau-Werte) die Farben kodiert werden.

Im Zeitschriften und Büchern wird das CMYK-Format verwendet, in dem Bilder aus einer Kombination der Farben Cyan, Magenta, Yellow und Black (CMYK) gedruckt werden.

Definieren Sie einen algebraischen Datentyp **Color** indem Sie Farben in verschiedenen Formaten darstellen, und definieren Sie damit die **rgb2cmyk** Funktion, die unter Verwendung folgender Formel und nach Eingabe der RGB-Werte die entsprechenden CMYK-Werte berechnet.

Wenn RGB = (0,0,0), dann ist CMYK = (0,0,0,1),

sonst werden die Werte nach folgenden Formeln berechnet:

$$w = \max(R/255, G/255, B/255)$$
$$C = (w - (R/255))/w$$
$$M = (w - (G/255))/w$$
$$Y = (w - (B/255))/w$$
$$K = 1 - w$$

7. Aufgabe (5 Punkte)

Nehmen wir an, wir müssen Berechnungen realisieren mit Zahlen, die folgende Form haben:

$a + b\sqrt{2}$, wobei ***a*** und ***b*** ganze Zahlen sind.

Würden wir zuerst die Wurzeln ausrechnen und dann die Summen machen, hätten wir Rundungsfehler, die im Laufe der Berechnungen größer werden können. Die Rundungsfehler können minimiert werden, wenn wir zuerst nur die ganzzahligen Operationen realisieren und das Ausrechnen der Wurzeln ans Ende verschieben. Beispiel:

$$\left(3 + 2\sqrt{2}\right) \cdot \left(2 + \sqrt{2}\right) = 10 + 7\sqrt{2}$$

a) Definieren Sie einen algebraischen Datentyp **RootNum**, der unsere Zahlen mit Hilfe der Koeffizienten **a** und **b** darstellt, und definieren Sie die Additions-, Subtraktions- und Multiplikationsoperation für diesen Datentyp.

b) Zum Testen definieren Sie eine Funktion **getValue,** die eine **RootNum** Variable in einer Gleitkommazahl ausrechnet.

8. Aufgabe (6 Punkte)

Ein Element einer Liste von ***n*** Objekten stellt die absolute Mehrheit der Liste dar,

wenn das Element mindestens $\left(\frac{n}{2} + 1\right)$-mal in der Liste vorkommt.

a) Definieren Sie eine **majority** Funktion, die das Majority-Element der Liste findet, wenn eines existiert oder sonst **Nothing** zurückgibt.

Die Signatur der Funktion soll wie folgt aussehen:

majority :: (Eq a) => [a] -> **Maybe** a

b) Analysieren Sie die Komplexität der Funktionsdefinition.

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) Geben Sie für alle Funktionen die entsprechende Signatur an.
6) Schreiben Sie getrennte Test-Funktionen für alle Aufgaben.
7) Die Lösungen sollen elektronisch (nur Whiteboard-Upload) abgegeben werden. **Keine verspätete Abgabe per Email ist erlaubt.**