WS 2020/2021

Funktionale Programmierung

4. Übungsblatt

Prof. Dr. Margarita Esponda

Ziel: Arbeiten mit Funktionen höherer Ordnung und endrekursiven Funktionen.

1. Aufgabe (3 Punkte)

Sei folgende Funktionsdefinition des BubbleSort-Algorithmus:

```
bubbleSort :: (Ord a) => [a] -> [a]
bubbleSort xs | isSorted (<=) xs = xs
              | otherwise = bubbleSort (moveBubble xs)
                where
                moveBubble [] = []
                moveBubble [x] = [x]
                moveBubble (x:y:rest) | (<=) x y   = x: moveBubble (y:rest)
                                      | otherwise  = y: moveBubble (x:rest)
```

Definieren Sie eine **traceBubbleSort-**Funktion, die nach jedem Aufruf der **moveBubble-**Funktion den Zwischenzustand der zu sortierenden Liste in einer Liste von Listen speichert, so dass am Ende des Sortieralgorithmus der Verlauf als Ergebnis angegeben wird.

Anwendungsbeispiele:

```
traceBubbleSort [0,1,3,8,0] => [[0,0,1,3,8], [0,1,0,3,8], [0,1,3,0,8], [0,1,3,8,0]]
```

2. Aufgabe (6 Punkte)

a) Die *Schwache Goldbachsche Vermutung* sagt, dass jede ungerade Zahl, die größer als **5** ist, als die Summe von **drei** Primzahlen geschrieben werden kann.

Definieren Sie eine Funktion **weakGoldbachTriples**, die bei Eingabe einer ungeraden Zahl die Liste aller *Schwachen Goldbachschen Tripel* ermittelt. Sie können in Ihrer Definition die Funktion **primzahlen** aus den Vorlesungsfolien verwenden.

Anwendungsbeispiel:

```
weakGoldbachTriples 19    =>    [(3,3,13),(3,5,11),(5,7,7)]
```

b) Definieren Sie eine Funktion, die die Vermutung bis zu einer eingegebenen Zahl **m** überprüft.

Anwendungsbeispiel:

```
wGTriplesUntil 300  =>  True
```

3. Aufgabe (4 Punkte)

a) Definieren Sie eine polymorphe Funktion **diffList**, die zwei Listen als Eingabe bekommt und jedes Element der zweiten Liste aus der ersten Liste entfernt, falls das Element in der ersten Liste vorhanden ist.

Anwendungsbeispiel:

diffList "Sebastian Meyer" "aaaaennn" => "Sbsti Myr"

b) Programmieren Sie mit Hilfe Ihre **diffList** Funktion eine Funktion, die aus einer beliebigen Liste natürlicher Zahlen die kleinste natürliche Zahl findet, die **nicht** in der Liste vorkommt.

Anwendungsbeispiel:

firstNatNotIn [3, 2, 0, 1, 8, 9, 12, 6] => 4

4. Aufgabe (4 Punkte)

Definieren Sie unter sinnvoller Verwendung der **foldl** Funktion die Funktion **toDecFrom**, die eine Zahl als Liste der einzelnen Ziffern in Basis **b** (mit **1<b<10**) bekommt und die Dezimaldarstellung der Zahl berechnet.

Anwendungsbeispiel: **toDecFrom** [1,3,2] 4 => 30

5. Aufgabe (7 Punkte)

Definieren Sie erneut die Funktion **flatten** :: [[a]] -> [a] aus Aufgabe **2)** des **3.**Übungsblatts.

a) Mit Hilfe der **foldr**-Funktion.

b) Schreiben Sie eine zweite Definition mit der **foldl**-Funktion.

c) Was sind die jeweiligen Vorteile und/oder Nachteile der Definitionen aus a) und b)?

6. Aufgabe (6 Punkte)

Eine **n** x **m** Matrix läßt sich in Haskell als Liste von Listen mit **n** Listen jeweils der Länge **m** (zeilenweise) oder **m** Listen jeweils der Länge **n** (spaltenweise) modellieren.

a) Definieren Sie eine polymorphe Haskell-Funktion, die zwei Matrizen addiert.

b) Die Multiplikation einer **n** x **m** mit einer **m** x **k** Matrix ergibt eine **n** x **k** Matrix. Definieren Sie die entsprechende Haskell-Funktion die das Ergebnis der Matrizenmultiplikation berechnet.

7. Aufgabe (4 Bonuspunkte)

Im Gegensatz zu den **foldl** und **foldr** Funktionen gibt es die **unfold** Funktion, die ein rekursives Pattern darstellt, um Listen zu produzieren. Betrachten Sie folgende Definition der **unfold** Funktion:

```
unfold p f g x | p x = []
              | otherwise = f x : unfold p f g (g x)
```

Definieren Sie erneut, unter Verwendung der **unfold**-Funktion, folgende Funktionen:

(map f), (iterate f) und dec2bin

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.**