WiSe 2015/2016

Funktionale Programmierung

3. Übungsblatt (Abgabe: Mo., den 09.11. um 10:10 Uhr)

Prof. Dr. Margarita Esponda

Ziel: Auseinandersetzung mit Listen und rekursiven Funktionsdefinitionen.

1. Aufgabe (7 Punkte)

Wir haben in der Vorlesung die Funktion **dec2bin** und **bin2dec** definiert (siehe Vorlesungsfolien), die die Binärdarstellung einer positive Dezimalzahl berechnet und das Ergebnis in einer Liste mit den einzelnen Binärziffern zurückgibt.

a) Definieren eine allgemeinere Funktion **fromDecTo**, die bei zusätzlicher Eingabe der Basis **b** mit **0<b<10** eine ganze natürliche Dezimalzahl in die entsprechende Zahl mit Basis **b** umrechnet.

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

b) Definieren Sie eine Funktion **toDecFrom**, die eine Zahl als Liste von einzelnen Ziffern in Basis **b** (mit **0<b<10**) bekommt und die Dezimaldarstellung der Zahl berechnet.

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

c) Schreiben Sie eine Variante der **toDecFrom** Funktion, die, ohne die Liste der Ziffern vorher umdrehen zu müssen, die gleiche Berechnung macht.

2. Aufgabe (5 Punkte)

Definieren Sie eine Haskell-Funktion, die aus einer beliebigen Binärzahl ***n*** (Zweierkomplement-Darstellung) die entsprechende negative Zahl (***-n***) berechnet. Es wird selbstverständlich angenommen, dass die Eingabe- und Ergebniszahl der Funktion immer die gleiche Bitlänge haben.

Anwendungsbeispiel: twoComplement [0,0,0,1,1,0,1,0] => [1,1,1,0,0,1,1,0]

3. Aufgabe (2 Punkte)

Verändern Sie die **balance**-Funktion aus den Vorlesungsfolien, sodass in einer beliebigen Zeichenkette kontrolliert wird, ob die Klammersetzung korrekt ist.

Anwendungsbeispiele: balanced "[(a+b+c)*[x-y]]/{(x+1)**0}" => True
balanced "[(a+b+c)*x-y)/{(x+1)*5}" => False

4. Aufgabe (6 Punkte)

Definieren Sie eine rekursive Funktion **deleteReps** , die wiederholte Zahlen aus einer Liste entfernt, so dass am Ende jede unterschiedliche Zahl nur einmal vorkommt.

Beispiel:

deleteReps [1,2,1,2,2,1,3,0] = [1,2,3,0]

5. Aufgabe (4 Punkte)

Schreiben Sie eine Funktion **chars2words**, die aus einem einfachen **String** die Liste aller Worte des Textes zurückgibt. Der Text besteht nur aus Buchstaben, Zahlen und Trennzeichen. Mit Trennzeichen sind Kommata, Punkte, Fragezeichen, Ausrufezeichen und Leerzeichen gemeint.

Die Funktion soll folgende Signatur haben:

chars2words :: [Char] -> [[Char]]

6. Aufgabe (4 Punkte)

In der Vorlesung wird die **randList**-Funktion besprochen, die in der Lage ist, bei Eingabe einer positiven Zahl **n** eine Liste mit **n** Pseudo-Zufallszahlen zu erzeugen.

Schreiben Sie eine Funktion **withoutReps**, die mit Hilfe der gleichen **random**-Funktion aus der Vorlesung bei Eingabe eines Startwertes (*seed*) die Anzahl der Pseudo-Zufallszahlen, bis eine Wiederholung vorkommt, berechnet.

7. Aufgabe (4 Punkte)

Definieren Sie eine Haskell-Funktion, die bei Eingabe einer Zahl in Oktal-Darstellung die Hexadezimal-Darstellung der Zahl berechnet.
Die Zahl soll als Zeichenkette eingegeben werden.

Anwendungsbeispiel: **okt2hex** "0770" => "1F8"

8. Aufgabe (4 Punkte)

Zur jeder Funktion (Aufgaben 1 bis 7) sollen Testfunktionen geschrieben werden.

Wichtige Hinweise:

In die Lösungen dieses Übungsblatts dürfen keine Listengeneratoren oder vordefinierte Funktionen höhere Ordnung verwendet werden.

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 Ihr Programme.

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

5) Schreiben Sie in allen Funktionen die entsprechende Signatur.