WS 2017/2018

Funktionale Programmierung

4. Übungsblatt

Prof. Dr. Margarita Esponda

(Abgabetermin: Mi. den 22. Nov., um 9:55)

Ziel: Auseinandersetzung mit Listengeneratoren und Funktionen höherer Ordnung.

1. Aufgabe (9 Punkte)

a) (2 P.) Definieren Sie eine Funktion **divisors**, die bei Eingabe einer natürlichen positiven Zahl **n** die Liste aller positive Teiler von **n** berechnet.

Anwendungsbeispiel:

divisors 250 => [1, 2, 5, 10, 25, 50, 125, 250]

b) (4 P.) Definieren Sie eine Funktion **trueDivisors**, die in der Lage ist eine unendliche Liste von ***(x,xs)*** Tupeln zu generieren, mit ***x*** gleich einer natürlichen Zahl ungleich ***0*** und ***xs*** gleich der Liste aller echten positiven Teiler von ***x***. Die Zahl **n** ist kein echter Teiler von **n** und soll deswegen nicht in die Ergebnisliste zurückgegeben werden.

Anwendungsbeispiel:

take 7 trueDivisors => [(1,[]),(2,[1]),(3,[1]),(4,[1,2]),(5,[1]),(6,[1,2,3]),(7,[1])]

c) (3 P.) Definieren Sie eine Funktion **primes**, die unter sinnvoller Anwendung Ihrer **trueDivisors** Funktion eine unendliche Liste von Primzahlen generiert.

Anwendungsbeispiel:

take 20 primes => [2,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53,59,61,67,71]

2. Aufgabe (6 Punkte)

Programmieren Sie eine Funktion **allFriendsSmaller**, die bei Eingabe einer natürlichen Zahl **n** alle befreundeten Zahlenpaare **(a, b)** berechnet, die kleiner **n** sind. D.h. mit **a<b<n**.

Zwei natürliche Zahlen **(m, n)** werden als "Befreundetes Zahlenpaar" bezeichnet, wenn jede Zahl gleich der Summe der echten Teiler der anderen Zahl ist. Definieren Sie zuerst eine Hilfsfunktion **friends**, die bei Eingabe zweier natürlicher Zahlen entscheidet, ob die Zahlen befreundet sind oder nicht.

Anwendungsbeispiel:

allFriendsSmaller 300 => [(220,284)]

3. Aufgabe (7 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 Triples* ermittelt. Sie können in Ihrer Definition die **primzahlen** Funktion 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** testet.

Anwendungsbeispiel:

wGTriplesUntil m 300 => True

4. Aufgabe (4 Punkte)

Definieren Sie eine polymorphe Funktion höherer Ordnung, die bei Eingabe eines Vergleichsoperators und einer Liste überprüft, ob die Liste nach dem eingegebenen Vergleichsoperator sortiert ist.

Anwendungsbeispiele:

isSorted (<=) [2, 2, 4, 5, 8, 9] => True
isSorted (<) [2, 2, 4, 5, 8, 9] => False
isSorted (==) [2, 2, 2, 3, 3, 3] => False

5. Aufgabe (4 Punkte)

Definieren Sie eine Funktion, die bei Eingabe einer natürlichen Zahl **n** die Liste aller (a, b, c) Pythagoras-Zahlentripel mit $a^2 + b^2 = c^2$, $0 < a \leq b \leq c \leq n$ und ohne Wiederholungen berechnet.

Anwendungsbeispiel:

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

6. Aufgabe (4 Punkte)

Definieren Sie eine polymorphe Funktion, 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.

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

Wichtige Hinweise:

2) Verwenden Sie geeignete Namen für Ihre Variablen und Funktionsnamen, die den semantischen Inhalt der Variablen oder die Semantik der Funktionen wiedergeben.

3) Verwenden Sie vorgegebene Funktionsnamen, falls diese angegeben werden.

4) Kommentieren Sie Ihre Programme.

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

5) Schreiben Sie in alle Funktionen die entsprechende Signatur.