SoSe 2014

Prof. Dr. Margarita Esponda

ProInformatik II: Funktionale Programmierung

5. Übungsblatt (6. Tag)

1. Aufgabe

Schreiben Sie eine Variante des Quicksort-Algorithmus, die eine Liste von Listen nach der Länge der Teillisten sortiert.

Anwendungsbeispiel:

qsort [[7,2,3], [10], [1,2], []] => [[], [10], [1,2], [7,2,3]]

2. Aufgabe

Analysieren Sie die Komplexität der **binSearch** Funktion aus der Vorlesung.

3. Aufgabe

Definieren Sie eine polymorphe Funktion **positions**, die die Positionen aller vorkommenden Elemente innerhalb einer Liste wiederum in einer Liste zurückgibt.

Anwendungsbeispiel: **positions** 'a' "Maria Antonieta" => [1, 4, 14]

a) Definieren Sie zuerst die Funktion unter Verwendung von expliziter Rekursion und Akkumulator-Technik.

b) Definieren Sie die Funktion unter sinnvoller Verwendung von Listengeneratoren und mindestens einer Funktion höherer Ordnung.

4. Aufgabe

Definieren Sie die Funktion **longestRepSeq**, die die längste sich wiederholende Objektsequenz aus einer Objekt-Liste findet (siehe Vorlesungsfolien). Die Funktion soll folgende Signatur haben.

longestRepSeq :: (Ord a) => [a] -> [a]

Anwendungsbeispiele: **longestRepSeq** "absdadsdahko" => "sda"

longestRepSeq [1,1,0,1,1,0,0,1,0,1,1] => [1,0,1,1]

Vorgehensweise:

a) Programmieren Sie zuerst folgende drei Hilfsfunktionen:

i) Eine Funktion **listOfSuffixes**, die alle verschiedenen Suffixe aus einer Zeichenkette in einer Liste als Ergebnisse zurückgibt.

Anwendungsbeispiel: **listOfSuffixes** "xyzab" => ["xyzab","yzab","zab","ab","b"]

ii) Definieren Sie eine Funktion **prefix**, die zwei Zeichenketten bekommt und das längste gemeinsame Prefix, falls es eines gibt, berechnet.

Anwendungsbeispiel: **prefix** "abcde" "abacde" => "ab"

iii) Definieren Sie eine Funktion **longestPrefix**, die eine Liste von Zeichenketten durchgeht und das längste Prefix aus zwei benachbarten Zeichenketten der Liste in einem Tupel als Ergebnis zurückliefert. Das erste Element des Tupels ist die Länge des Prefixes.

Anwendungsbeispiel:

longestPrefix ["a","abca","bca","bcadabca","ca","cdabca"] => (3,"bca")

b) Programmieren Sie dann unter Verwendung Ihrer Funktionen in i), ii), iii) und ein geeignete Sortieralgorithmus die **longestRepSeq** Funktion. Verwenden Sie Funktionskomposition in Ihrer Definition.

c) Analysieren Sie die Komplexität Ihrer Funktionen.

Wichtige Hinweise:

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

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

5) Kommentieren Sie Ihre Programme.

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

5) Schreiben Sie in alle Funktionen die entsprechende Signatur.