TECHNISCHE UNIVERSITÄT
CHEMNITZ

Fakultät für Informatik
Professur Praktische Informatik
Prof. Dr. G. Rünger
Dr. R. Dietze

Funktionale Programmierung/Höhere Programmiersprachen

Wintersemester 2022/2023

3. Übungsserie

Falls Sie eine Prüfungsvorleistung erbringen müssen, geben Sie Ihre Lösungen bitte **jeweils vor Beginn der ersten Übungseinheit (Donnerstag 15:30 Uhr)** beim Übungsleiter ab oder laden sie bis zu diesem Zeitpunkt unter dem Punkt Abgabe im OPAL hoch.

Aufgabe 1:

Geben Sie jeweils eine Haskell-Funktion an, die

a) ein gegebenes Integer-Tripel in aufsteigende Reihenfolge sortiert.
`orderTriple :: (Int, Int, Int) -> (Int, Int, Int)`

b) ein gegebenes Integer-Quadrupel um eine gegebene positive Anzahl an Positionen nach rechts (negative Anzahl nach links) rotiert.
`rotateQuadruple :: (Int, Int, Int, Int) -> Int -> (Int, Int, Int, Int)`
Beispiel: `rotateQuadruple (1,2,3,4) 1 = (4,1,2,3)`
`rotateQuadruple (1,2,3,4) (-5) = (2,3,4,1)`

c) aus einer Liste von Integer-Paaren das Paar mit der maximalen Summe der Komponenten zurückliefert. Bei einer leeren Liste soll (0,0) zurückgegeben werden.
`maxpairs :: [(Int, Int)] -> (Int, Int)`
Beispiel: `maxpairs [(2,1),(3,2),(5,3),(2,2)] = (5,3)`

Aufgabe 2:

Geben Sie Haskell-Funktionen mit folgenden Funktionalitäten an.

a) Funktion `scalar` zur Berechnung des Skalarprodukts zweier Vektoren, die durch Integer-Listen gegeben sind. Werden Listen unterschiedlicher Länge übergeben, soll eine Fehlermeldung ausgegeben werden.
Beispielaufruf: `scalar [1..10] [2,4..20] = 770`
`scalar [1,2,3] [5,7] = Program error: Vektoren ungleich lang`

b) Funktion `isPalin` zur Ermittlung ob eine gegebene Liste von Integern ein Palindrom ist (d. h. von vorn und hinten gelesen gleich ist).
Beispielaufruf: `isPalin [1,2,3,2,1] = True`
`isPalin [1..7] = False`

Geben Sie für die nachfolgenden Haskell-Funktionen jeweils eine Variante **ohne** und eine **mit** List comprehension an.

c) Funktion `matches`, die alle Vorkommen eines Integers aus einer Liste in einer neuen Liste zurückliefert.
Beispielaufruf: `matches [1,4,2,1,4,1,6,3] 1 = [1,1,1]`
`matches [1,2,3,4,5] 6 = []`

Aufgabe 3:

Geben Sie Haskell-Funktionen mit folgenden Funktionalitäten an.

a) Funktion `mergeSort` zum Sortieren einer Liste von Integer-Werten in aufsteigender Reihenfolge mit dem Sortierverfahren Mergesort.
Hinweis: Die Funktion `take::Int->[a]->[a]` gibt die ersten n Einträge einer Liste zurück. Die Funktion `drop::Int->[a]->[a]` entfernt die ersten n Einträge einer Liste.

b) Funktion `nor3` zur Berechnung des negierten logischen Oder für drei boolesche Werte **ohne Verwendung Boolescher Operatoren**. Geben Sie zwei Varianten unter Verwendung von **Guarded commands** und **Pattern matching** an.

Aufgabe 4:

Zeigen Sie mittels Equational reasoning, dass folgende Aussage wahr ist:

Der Haskell-Ausdruck `length(1:2:[] ++ [3,4]) != 4` ist `False`.

Hinweis: Nutzen Sie die, in der Vorlesung angegebenen Eigenschaften des Operators `:` sowie die Definitionen des Operators `++` und der Funktion `length`.