TECHNISCHE UNIVERSITÄT
CHEMNITZ

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

# Funktionale Programmierung/Höhere Programmiersprachen

Wintersemester 2022/2023

4. 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**:

Zeigen Sie mittels Equational reasoning, dass der Haskell-Ausdruck
`[sum (dbl xs)] == dbl [sum xs]`
für eine Liste von Integer-Werten `xs` wahr ist. Die Funktionen `sum` und `dbl` sind wie folgt definiert:

```
sum :: [Int] -> Int
sum [] = 0
sum x:xs = x + sum xs

dbl :: [Int] -> [Int]
dbl [] = []
dbl x:xs = (2*x):(dbl xs)
```

**Hinweis:** Führen Sie einen Beweis mittels Induktion über die Liste von Integer-Werten durch. Nutzen Sie die, in der Vorlesung gegebenen Eigenschaften des cons-Operators `:`.

**Aufgabe 2**:

Geben Sie Haskell-Funktionen mit folgenden Funktionalitäten unter Verwendung Funktionen höherer Ordnung an.

**a)** Vertauschen der Reihenfolge der Argumente einer übergebenen Funktion.
`flip :: (a -> b -> c) -> (b -> a -> c)`
Beispiel: `flip (-) 3 4 = 1`
`flip mod 3 17 = 2`

**b)** Eine Funktion `total :: (Int -> Int) -> (Int -> Int)`, so dass `total f` eine Funktion zurückliefert, die für ein übergebenes `n` den Wert `f 0 + f 1 + f 2 + ... + f n` berechnet.
Beispiel: `total (+2) 1 = 5`
`total id 100 = 5050`

**c)** Eine Funktion `composeList`, die eine Liste von Funktionen `f :: a-> a` in eine einzige Funktion `(a -> a)` umwandelt, die alle Funktionen der Liste nacheinander ausführt. Bei Übergabe einer leeren Liste soll die Identitätsfunktion `id` genutzt werden.
Beispiel: `composeList [(*2),(+2)] 1 = 4`
`composeList [(++"Hallo"),(++" "),(++"Welt"),(++"!")] [] = "Hallo Welt!"`

**d)** Berechnung einer numerischen Approximation des Integrals einer übergebenen Funktion `f`. Die erzeugte Funktion soll den Flächeninhalt unterhalb des Graphen von `f` zwischen zwei Punkten berechnen können.
`integrate :: (Float -> Float) -> (Float -> Float -> Float)`
Beispiel:
```
integrate id 0 5 = 12.5
integrate (^2) 1 2 = 2.33333
```

**e)** Funktionsauswertungen der als Liste gegebenen Funktionen mit zwei vorgegebenen Float-Werten.
`calcOps :: [(Float -> Float -> Float)] -> Float -> Float -> [Float]`

**f)** Lineare Suche nach dem ersten Auftreten eines Elements in einer Liste anhand einer vorgegebenen Prädikatsfunktion. Zurückgegeben werden soll die Listenposition oder `-1` falls kein entsprechendes Element in der Liste vorhanden ist.
`linearSearch :: (a -> Bool) -> [a] -> Int`

**g)** Generisches Sortierverfahren `quickSortGen` basierend auf Quicksort (siehe Vorlesung), welches eine beliebige Vergleichsfunktion verwendet und Funktion `quickSortStrings` (basierend auf `quickSortGen`) zur alphabetischen Sortierung einer Liste von Strings.