Dieses Dokument enthält Fragen, kleine Aufgaben und andere Ressourcen zum Thema Deklarative Programmierung. Die Inhalte dieses Dokuments sollen dir helfen, dein Verständnis über Haskell und Prolog zu prüfen.

Funktionale Programmierung

Test 1 Was bedeutet es, wenn eine Funktion keine Seiteneffekte hat?

Test 2 Haskell ist eine streng getypte Programmiersprache. Was bedeutet das?

Test 3 Wenn du eine Schleife in Haskell umsetzen möchtest, auf welches Konzept musst du dann zurückgreifen?

Test 4 In imperativen Programmiersprachen sind Variablen Namen für Speicherzellen, deren Werte zum Beispiel in Schleifen verändert werden können. Als Beispiel betrachte

```
def clz(n):
  k = 0
  while n > 0:
    n /= 2
    k += 1
  return 64 - k
```

In Haskell sind Variablen keine Namen für Speicherzellen. Wie können wir dieses Programm in Haskell umsetzen? Wo wandert das `k` hin?

Test 5 Auf was müssen wir achten, wenn wir eine rekursive Funktion definieren? Die Antwort ist abhängig von dem, was die Funktion berechnen soll. Denke über die verschiedenen Möglichkeiten nach.

Test 6 Gegeben sei das folgende Haskell-Programm.

```
even :: Int -> Bool
even 0 = True
even n = odd (n - 1)

odd :: Int -> Bool
odd 0 = False
odd n = even (n - 1)
```

- Berechne das Ergebnis von `odd 1` händisch.
- Wie sieht der Auswertungsgraph für den Ausdruck `odd 1` aus?
  - Welcher Pfad entspricht deiner händischen Auswertung?
  - Welcher Pfad entspricht der Auswertung wie sie in Haskell stattfindet?
  - Wie sieht es mit Python aus?

Test 7 Es wird als sauberer Programmierstil angesehen, Hilfsfunktionen, die nur für eine Funktion relevant sind, nicht auf der höchsten Ebene zu definieren. Mithilfe welcher Konstrukte kannst du diese lokal definieren?

Test 8 Das Potenzieren einer Zahl x (oder eines Elements einer Halbgruppe) mit einem natürlich-zahligen Exponent n ist in $\mathcal{O}(\log n)$ Laufzeit möglich[1]. Dafür betrachten wir

$$x^n = \begin{cases} \left(x^{\frac{n}{2}}\right)^2 & \text{falls } n \text{ gerade} \\ x \cdot x\left(x^{\frac{n}{2}}\right)^2 & \text{sonst} \end{cases}$$

[1] Binäre Exponentiation – Wikipedia

Implementiere eine Funktion, die diese Variante des Potenzierens umsetzt.

Test 9 Gegeben ist folgender Ausdruck.

```
let v = 3
    w = 5
    x = 4
    y = v + x
    z = x + y
 in y
```

Welche Belegungen der Variablen werden tatsächlich berechnet, wenn wir `y` ausrechnen?

Test 10 Wie werden algebraische Datentypen in Haskell definiert?

Test 11 Was ist charakterisierend für Aufzählungstypen, einen Verbundstypen und einem rekursiven Datentypen? Gebe Beispiele für jeden dieser Typarten an.

Test 12 Geben ist der Typ `IntList` mit `data IntList = Nil | Cons Int IntList`. Weiter kann mithilfe der Funktion

```
lengthIntList :: IntList -> Int
lengthIntList Nil         = 0
lengthIntList (Cons _ xs) = 1 + lengthIntList xs
```

die Länge einer solchen Liste berechnet werden. Du möchtest nun auch die Längen von Listen berechnen, die Buchstaben, Booleans oder Gleitkommazahlen enthalten. Was stört dich am bisherigen Vorgehen? Kennst du ein Konzept mit dessen Hilfe du besser an dein Ziel kommst?

Test 13 Wie ist die Funktion `lengthIntList :: IntList -> Int` aus dem vorherigen Test definiert?

Test 14 Du hast einen Datentypen definiert und möchtest dir Werte des Typen nun z.B. im GHCi anzeigen lassen. Was kannst du tun, um an dieses Ziel zu kommen?

Wenn du auf der Suche nach weiteren Übungsaufgaben bist, mit denen du deine Programmierkenntnisse in Prolog verbessern möchtest, bietet sich die Liste P-99: Ninety-Nine Prolog Problems an. Lösungen sind ebenso auf der Seite verfügbar. Für Haskell gibt es eine ähnliche Seite H-99: Ninety-Nine Haskell Problems.

Weitere Links:

- Haskelite: Ein Schritt-für-Schritt Interpreter für (eine Teilmenge von) Haskell
- Haskell Cheatsheet