# 4. Übungsblatt zu Programmierung 1, WS 2010/11

Prof. Dr. Gert Smolka, Christian Doczkal, M.Sc.
www.ps.uni-sb.de/courses/prog-ws10/

Lesen Sie im Buch: Kapitel 4

**Aufgabe 4.2** Betrachten Sie den Ausdruck 1::2::*nil* @ 3::4::*nil*.

a) Geben Sie die Baumdarstellung des Ausdrucks an.
b) Geben Sie die Baumdarstellung der beschriebenen Liste an.
c) Geben Sie die beschriebene Liste mit „[...]“ an.

**Aufgabe 4.4 (Enum)** Schreiben Sie mithilfe der Prozedur *iterdn* (§ 3.13) eine Prozedur *enum* : *int* → *int* → *int list*, die zu zwei Zahlen $m \leq n$ die Liste $[m, \ldots, n]$ liefert. Beispielsweise soll *enum* 3 6 = [3, 4, 5, 6] gelten. Für $m > n$ soll *enum* die leere Liste liefern.

**Zusatzaufgabe Z4.1** Schreiben Sie eine polymorphe Prozedur *member* : *''a* → *''a list* → *bool* die testet, ob ein Wert als Element in einer Liste vorkommt. Lösen Sie dies auf drei Arten:

a) durch eine regelbasierte Prozedurdeklaration (formulieren Sie zunächst passende Rekursionsgleichungen),
b) mithilfe der vordeklarierten Prozedur *List.exists*,
c) mithilfe der Prozedur *foldl*.

**Aufgabe 4.10 (Count)** Schreiben Sie mithilfe von *foldl* eine polymorphe Prozedur *count* : *''a* → *''a list* → *int*, die zählt, wie oft ein Wert in einer Liste als Element vorkommt. Beispielsweise soll *count* 5 [2, 5, 3, 5] = 2 gelten.

**Aufgabe 4.11 (Dezimaldarstellung)** Die Dezimaldarstellung einer natürlichen Zahl ist die Liste ihrer Ziffern. Beispielsweise hat 7856 die Dezimaldarstellung [7, 8, 5, 6].

a) Deklarieren Sie eine Prozedur *dec* : *int* → *int list*, die die Dezimaldarstellung einer natürlichen Zahl liefert. Verwenden Sie *div* und *mod*.
b) Deklarieren Sie mithilfe von *foldl* eine Prozedur *num* : *int list* → *int*, die zu einer Dezimaldarstellung die dargestellte Zahl liefert.

**Aufgabe 4.12** Deklarieren Sie die Faltungsprozedur *foldr* mithilfe der Faltungsprozedur *foldl*. Verwenden Sie dabei keine weitere rekursive Hilfsprozedur.

**Aufgabe 4.13** Deklarieren Sie die Faltungsprozedur *foldl* mithilfe der Faltungsprozedur *foldr*. Gehen Sie wie folgt vor:

a) Deklarieren Sie *append* mithilfe von *foldr*.
b) Deklarieren Sie *rev* mithilfe von *foldr* und *append*.
c) Deklarieren Sie *foldl* mithilfe von *foldr* und *rev*.
d) Deklarieren Sie *foldl* nur mithilfe von *foldr*.

**Aufgabe 4.15 (Last)** Deklarieren Sie eine Prozedur $last : \alpha\ list \to \alpha$, die das Element an der letzten Position einer Liste liefert. Wenn die Liste leer ist, soll die Ausnahme *Empty* geworfen werden.

**Aufgabe 4.16 (Max)** Schreiben Sie mit *foldl* eine Prozedur $max : int\ list \to int$, die das größte Element einer Liste liefert. Wenn die Liste leer ist, soll die Ausnahme *Empty* geworfen werden. Verwenden Sie die vordefinierte Prozedur $Int.max : int * int \to int$.

**Aufgabe 4.17 (Take und Drop)** Schreiben Sie zwei polymorphe Prozeduren *take* und *drop*, die gemäß $\alpha\ list * int \to \alpha\ list$ getypt sind und die folgende Spezifikation erfüllen:

a) $take(xs, n)$ liefert die ersten $n$ Elemente der Liste $xs$. Falls $n < 0$ oder $|xs| < n$ gilt, soll die Ausnahme *Subscript* geworfen werden.
b) $drop(xs, n)$ liefert die Liste, die man aus $xs$ erhält, wenn man die ersten $n$ Elemente weglässt. Falls $n < 0$ oder $|xs| < n$ gilt, soll die Ausnahme *Subscript* geworfen werden.

Hinweis: Verwenden Sie *orelse* und die Prozeduren *hd*, *tl* und *null*.

**Aufgabe 4.19** Entscheiden Sie für jeden der folgenden Werte, ob er das Muster $(x,\ y :: \_ :: z,\ (u, 3))$ trifft. Geben Sie bei einem Treffer die Bindungen für die Variablen des Musters an.

a) $(7,\ [1],\ (3, 3))$
b) $([1, 2],\ [3, 4, 5],\ (11, 3))$

**Aufgabe 4.21** Schreiben Sie eine Prozedur $size : string \to int$, die die Länge eines Strings liefert (z.B. *size* "*hut*" = *3*).

**Aufgabe 4.22** Schreiben Sie eine Prozedur $reverse : string \to string$, die Strings reversiert (z.B. *reverse* "*hut*" = "*tuh*").

**Aufgabe 4.23** Schreiben Sie eine Prozedur $isDigit : char \to bool$, die mithilfe der Prozedur *ord* testet, ob ein Zeichen eine der Ziffern $0, \ldots, 9$ ist. Nützen Sie dabei aus, dass *ord* die Ziffern durch aufeinander folgende Zahlen darstellt.

**Aufgabe 4.24** Schreiben Sie eine Prozedur $toInt : string \to int$, die zu einem String, der nur aus Ziffern besteht, die durch ihn dargestellte Zahl liefert (z.B. *toInt* "*123*" = *123*). Falls der String Zeichen enthält, die keine Ziffern sind, soll die Ausnahme *Domain* geworfen werden.

**Aufgabe 4.25** Schreiben Sie mithilfe der Prozeduren *explode* und *ord* eine Prozedur $less : string \to string \to bool$, die für zwei Strings $s$, $s'$ dasselbe Ergebnis liefert wie $s < s'$.

**Zusatzaufgabe Z4.2** Schreiben Sie mithilfe der Prozedur *iter* eine Prozedur $tab : int \to (int \to \alpha) \to \alpha\ list$, die für Argumente $n$ und $f$ das Ergebnis $[f\ 0, \ldots, f\ (n - 1)]$ liefert.