4. Übungsblatt zu Programmierung WS 2002 / 03

Prof. Dr. Gert Smolka und Dipl.-Inform. Thorsten Brunklaus
www.ps.uni-sb.de/courses/prog-ws02/

Abgabe: Montag, 18. November 2002

Alle Aufgaben sind **ausschließlich** per E-Mail an Ihren Übungsgruppenleiter abzugeben. Ihre Abgaben können nur berücksichtigt werden, wenn Sie jeweils Ihren Namen und **die Nummer Ihrer Übungsgruppe angeben**.

Aufgabe 4.1: Last (8) Schreiben Sie eine Prozedur

```
last : 'a list -> 'a
```

die das letzte Element einer Liste liefert. Wenn die Liste leer ist, soll die Ausnahme `Empty` geworfen werden.

Aufgabe 4.2: Enum (8) Schreiben Sie eine Prozedur

```
enum : int * int -> int list
```

die für zwei Zahlen $m \leq n$ die Liste $[m, m+1, \ldots, n]$ liefert. Für $m > n$ soll `enum` die leere Liste liefern.

Aufgabe 4.3: Nth, Take, Drop ($15 = 3 * 5$) Schreiben Sie drei Prozeduren

```
nth  : 'a list * int -> 'a
take : 'a list * int -> 'a list
drop : 'a list * int -> 'a list
```

wie folgt:

- `nth(`*xs*`,`*n*`)` liefert das *n*-te Element der Liste *xs*. Dabei sollen die Elemente von *xs* mit 0 beginnend nummeriert sein. Falls $n < 0$ oder $length(xs) \leq n$ gilt, soll die Ausnahme `Subscript` geworfen werden.
- `take(`*xs*`,`*n*`)` liefert die ersten *n* Elemente der Liste *xs*. Falls $n < 0$ oder $length(xs) < n$ gilt, soll die Ausnahme `Subscript` geworfen werden.
- `drop(`*xs*`,`*n*`)` liefert die Liste, die man aus *xs* erhält, wenn man die ersten *n* Elemente weglässt. Falls $n < 0$ oder $length(xs) < n$ gilt, soll die Ausnahme `Subscript` geworfen werden.

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

Aufgabe 4.4: Max (10) Schreiben Sie mit `foldl` eine Prozedur

```
max : int list -> int
```

die das größte Element einer Liste liefert. Wenn die Liste leer ist, soll die Ausnahme `Empty` geworfen werden.

Aufgabe 4.5: Member ($15 = 3 * 5$) Sie sollen drei Varianten einer Prozedur

```
member : ''a -> ''a list -> bool
```

schreiben, die testet ob ein Wert in einer Liste vorkommt.

(a) Die erste Variante soll keine andere Prozedur benutzen.

(b) Die zweite Variante soll mit `List.exists` und ohne Rekursion geschrieben werden.

(c) Die dritte Variante soll mit `foldl` und ohne Rekursion geschrieben werden.

Aufgabe 4.6: Count (10) Schreiben Sie mit `foldl` eine Prozedur

```
count : ''a -> ''a list -> int
```

die für einen Wert die Anzahl seiner Auftreten in einer Liste berechnet. Beispielsweise liefert `count 5 [1,2,3,5,4,5]` das Ergebnis 2.

Aufgabe 4.7: Dezimaldarstellung ($10 = 2 * 5$) Die Dezimaldarstellung einer natürlichen Zahl ist die Liste ihrer Ziffern. Beispielsweise hat 7856 die Dezimaldarstellung [7, 8, 5, 6].

(a) Schreiben Sie eine Prozedur

```
dec : int -> int list
```

die die Dezimaldarstellung einer natürlichen Zahl berechnet. Verwenden Sie `div` und `mod`.

(b) Schreiben Sie mithilfe von `foldl` eine Prozedur

```
int : int list -> int
```

die zu einer Dezimaldarstellung die dargestellte natürliche Zahl berechnet.

Aufgabe 4.8: Permutationen (4) Sei eine Sortierprozedur

```
sort : int list -> int list
```

gegeben. Schreiben Sie damit eine Prozedur

```
perm : int list * int list -> bool
```

die testet, ob zwei Listen bis auf Permutation gleich sind.

Aufgabe 4.9: Partition ($10 = 2 * 5$) Schreiben Sie zwei Varianten einer Prozedur

```
partition : int -> int list -> int list * int list
```

die zu einer Zahl *x* und einer Liste *xs* zwei Listen *us* und *vs* wie folgt berechnet:

(a) *us@vs* ist eine Permutation von *xs*.

(b) Alle Elemente von *us* sind echt kleiner als *x* und alle Elemente von *vs* sind größer gleich als *x*.

Die erste Variante soll keine andere Prozedur benutzen. Die zweite Variante soll mit `foldl` und ohne Rekursion geschrieben werden. Verwenden Sie dabei eine Hilfprozedur des Typs

```
int * (int list * int list) -> int list * int list
```

Aufgabe 4.10: Quicksort (10) Quicksort ist ein klassischer Sortieralgorithmus, der auf der folgenden Beobachtung beruht.
Seien $x \in \mathbb{Z}$ und $xs, us, vs \in \mathcal{L}(\mathbb{Z})$ wie folgt:

(a) $us@vs$ ist eine Permutation von xs.

(b) Alle Elemente von us sind echt kleiner als x und alle Elemente von vs sind größer gleich als x.

Weiter sei *sort* die Sortierfunktion $\mathcal{L}(\mathbb{Z}) \to \mathcal{L}(\mathbb{Z})$. Dann gilt die Gleichung

$$sort(x :: xs) \;=\; sort(us) \;@\; [x] \;@\; sort(vs)$$

Schreiben Sie mithilfe dieser Gleichung und der Prozedur `partition` aus Aufgabe 4.9 eine Sortierprozedur `qsort : int list -> int list`.