

## Übungen zu Informatik I

### Aufgabe 14-1 Wiederholung: Typisierung (keine Abgabe)

Geben Sie die Typen folgender SML-Funktionen an:

- a) `fun add1 n = n + 1`
- b) `fun add2 x = x + 2.0`
- c) `fun rev nil = nil`  
`| rev (x::xs) = (rev xs) @ [x]`
- d) `fun addl(n,nil) = nil`  
`| addl(n,m::ms) = (n+m)::(addl(n,ms))`
- e) `fun mappend f [] = []`  
`| mappend f (x::xs) = (f x) @ (mappend f xs);;`

### Aufgabe 14-2 Wiederholung: Additive Terme (keine Abgabe)

Die Menge *Term* der *additiven Terme* über den ganzen Zahlen  $\mathbb{Z}$  ist induktiv definiert wie folgt:

1. *Term* enthält die ganzen Zahlen  $\mathbb{Z}$
  2. Sind  $t_1, t_2 \in \text{Term}$ , so sind auch  $\text{Sum}(t_1, t_2) \in \text{Term}$  und  $\text{Dif}(t_1, t_2) \in \text{Term}$ .
- a) Geben Sie eine `datatype`-Deklaration *Term* an, die additive Terme in SML repräsentiert.
  - b) Definieren Sie eine SML-Funktion  $\text{eval} : \text{Term} \rightarrow \mathbf{int}$ , die Terme auswertet. Dabei sollen  $\text{Sum}(t_1, t_2)$  als Addition von  $t_1$  und  $t_2$  und  $\text{Dif}(t_1, t_2)$  als Differenz von  $t_1$  und  $t_2$  interpretiert werden.

### Aufgabe 14-3 Addition (keine Abgabe)

Wir betrachten die folgende (ungeschickte) Definition einer Additionsfunktion:

```
fun add(x, y) = if x = 0 then y else add(x-1, y) + 1;
```

- a) Geben Sie das zu *add* gehörende Fixpunktfunktional *F* an.
- b) Geben Sie eine explizite Darstellung der Approximationen  $F^i(\Omega)$  für  $i \geq 0$  an.
- c) Zeigen Sie: Für  $f = \text{fix}(F)$  gilt:  $f(x, y) = x + y$ .

### Aufgabe 14-4 Imperative Programmierung (keine Abgabe)

Schreiben Sie iterative Algorithmen *potit*, *istquadratit*, *teilersummeit*, *mapit*, *foldlit* in der in der Vorlesung eingeführten Pseudocode-Notation, die jeweils folgendes leisten:

- a) Die Funktion *potit* berechnet für eine reelle Zahl  $x$  und eine natürliche Zahl  $k$  die  $k$ -te Potenz von  $x$ .
- b) Die Funktion *istquadratit* bestimmt für eine natürliche Zahl, ob sie eine Quadratzahl ist.
- c) Die Funktion *teilersummeit* bestimmt für eine natürliche Zahl  $n$  die Summe aller Teiler  $i \geq 0$  von  $n$ .

- d) Für eine Funktion  $f : \alpha \rightarrow \beta$  und eine Liste  $l : \alpha$  **list** liefert ein Aufruf von  $mapit(f, l)$  die Liste, die entsteht, wenn man  $f$  elementenweise auf  $l$  anwendet.
- e) Die Funktion  $foldlit$  berechnet die Links-Faltung einer Liste mit einer Funktion, d. h. für eine Funktion  $f$  vom Typ  $\alpha \times \beta \rightarrow \beta$ , eine Liste  $l = [x_1, \dots, x_n]$  vom Typ  $\alpha$  **list** und ein Element  $z$  vom Typ  $\beta$  gilt

$$foldlit(f, l, z) = f(x_n, f(x_{n-1}, \dots, f(x_1, z) \dots))$$