Technische Universität München
Fakultät für Informatik

A. Berlea
berlea@in.tum.de
Raum: MI 02.07.053

Programmiersprachen

Wintersemester 2006/2007

2. Übungsblatt *9. November 2006*

Aufgabe 1:

Definiere folgende Funktionen!

a) `null : 'a list -> bool`: testet ob eine Liste leer ist.

b) `hd : 'a list -> 'a`: liefert das erste Element einer Liste zurück.

c) `tl : 'a list -> 'a list`: liefert die Eingabeliste ohne ihr erstes Element zurück.

d) `last : 'a list -> 'a`: liefert das letzte Element einer Liste zurück.

e) `nth : 'a list * int -> 'a` : liefert das n-te Element einer Liste zurück.

f) `take : 'a list * int -> 'a list`: liefert die ersten n Elemente einer Liste zurück.

g) `drop : 'a list * int -> 'a list`: liefert die Eingabeliste ohne ihre ersten n Elemente zurück.

h) `length : 'a list -> int`: berechnet die Länge einer Liste.

i) `rev : 'a list -> 'a list`: invertiert eine Liste.

j) `append : 'a list * 'a list -> 'a list`: konkateniert zwei Listen.

k) `zip : 'a list * 'b list -> ('a * 'b) list`: erzeugt aus zwei Listen eine Liste von Paaren mit dem ersten Element aus der ersten und dem zweiten aus der zweiten Eingabeliste.

l) `unzip : ('a * 'b) list -> 'a list * 'b list`: die Inverse der `zip` Funktion.

m) `sum : int list -> int`: summiert die Element einer Liste von `int`s auf.

n) `min : int list -> int`: berechnet das Minimum einer List.

Aufgabe 2:

Mergesort ist ein rekursiver Sortieralgorithmus, der gemäß dem Divide-and-Conquer-Prinzip arbeitet. Hier soll Mergesort zum Sortieren von Listen implementiert werden. Die Arbeitsweise des Algorithmus läßt sich wie folgt skizzieren.

- Null-/und einelementige Listen sind sortiert
- Listen mit mehr als einem Element werden in zwei möglichst gleich große Teillisten aufgeteilt. Die Teillisten werden durch einen rekursiven Aufruf von Mergesort sortiert und anschließend in einem Mischschritt zu einer sortierten Liste zusammengesetzt.

Definiere eine Funktion `mergesort : 'a list -> 'a list`, die den skizzierten Algorithmus implementiert. Folgende Funktionen sollten dazu zunächst definiert werden:

- Die Funktion `split : 'a list -> 'a list * 'a list` soll eine Liste in zwei möglichst gleich große Teillisten aufteilen.
- Die Funktion `merge : 'a list * 'a list -> 'a list` soll den Mischschritt implementieren. Als Argument erhält sie ein Paar `(l1, l2)` von bereits sortierten Listen. Als Ergebnis liefert sie die sortierte Liste aller Elemente aus `l1` und `l2` zurück.

Aufgabe 3:

Berechne den Typ der folgenden Ausdrücke:

a)
```
fn x => (fn y => x + y)
```

b)
```
fun sum x = x + sum (x-1)
```

c)
```
fun m f =
  fn l =>
    case l of
      nil => nil
    | x::r => (f x)::((m f) r)
```

d)
```
fun f p =
  fn l =>
    case l of
      nil => nil
    | x::r => if (p x) then x::((f p) r)
              else (f p) r
```