

Übungen zu Informatik I (Lösungsvorschlag)

Aufgabe 9-1 Map und Faltung für Reihungen (keine Abgabe)

Folgende SML-Datei enthält die Lösung zu allen Teilaufgaben.

```
use "vect.sml";

(* (a) Mapvect *)
(* Es empfiehlt sich, eine allgemeinere Funktion "mapvecti" zu
   definieren, die zwei Argumente übergibt: Den Index des aktuellen
   Elements und den Wert. "mapvect" entsteht dann aus "mapvecti",
   indem man den Index ignoriert. *)

fun mapvect_emb f i v v' = if i > dim v
  then v'
  else mapvect_emb f
    (i+1)
    v
    (update(v',i,f(i,get(v,i))));;
fun mapvecti f v = mapvect_emb f 1 v (init(dim v, f(1,get(v,1))));;
fun mapvect f v = mapvecti (fn (i,a) => f a) v;;

(* (b) Konstruktoren *)

fun finit (l,f) = mapvecti (fn (i,v) => f i) (init(l,f(1))));;
fun linit l = finit (length l, fn i => List.nth(l, (i-1))));;

val v1 = linit [1,2,3,4];;
val v2 = linit (explode "asdf");;
val v3 = linit [1.0, 2.0, 3.0];;
val v4 = linit [7.0, 3.0, 5.2];;
val v5 = linit [81.4, 38.23, 93.3];;

(* (c) Linksfaltung *)

fun foldlvect_emb i f e v = if dim v = 0
  then e
  else if i = dim v
    then f (get(v,i), e)
    else foldlvect_emb (i+1) f (f (get(v,i), e)) v;;
fun foldlvect f e v = foldlvect_emb 1 f e v;;

(* (d) Skalarprodukt *)
```

```

fun curry f x y = f(x,y);;
val mul : real * real -> real = (op * );;

fun sprod x v = mapvect (curry mul x) v;;

```

(* (e) Vektornorm *)

```

fun norm v = Math.sqrt (foldlvect (op +)
  0.0
  (mapvecti (fn (i,x) => let val xi = get(v,i)
    in xi * xi end)
  v));;

```

Aufgabe 9-2

Faltung und Map für Bäume

(keine Abgabe)

Folgende SML-Datei enthält die Lösung zu allen Teilaufgaben.

```

use "bintree.sml";

```

(* (a) Faltung für Bäume *)

```

fun folds f s b =
  if isempty b then s
  else f( folds f s (left b), (root b), folds f s (right b));

```

(* (b) Quantoren für Bäume *)

```

fun allbintree p = folds (fn (l, m, r) => l andalso p(m) andalso r) true;
fun existsbintree p = folds (fn (l, m, r) => l orelse p(m) orelse r) false;

```

(* (c) Mapbintree *)

```

fun mapbintree f b = if isempty b
  then empty
  else build (f(root(b)),
    mapbintree f (left b),
    mapbintree f (right b));

```

Aufgabe 9-3

Reihungen

(6 Punkte)

Folgende SML-Datei enthält die Lösung zu allen Teilaufgaben.

```

use "vect.sml";

```

(* (a) Umkehren eines Vektors. *)

```

fun reversevect v = linit (foldlvect (op ::) nil v);;

```

(* (b) Zipvect *)

```

exception differentlength;;

fun zipvect v1 v2 =
  if dim v1 = dim v2
  then mapvecti (fn (i,a) => (get(v1,i),get(v2,i)))
  (init(dim v1, (get(v1, 1),get(v2,1))))
  else raise differentlength;;

(* (c) inneres Produkt *)

fun intprod v1 v2 = mapvect mul (zipvect v1 v2);;

(* (d) Vektorprodukt *)

exception wronglength

fun crossprod (v1:real vect) v2 =
  if dim v1 <> 3 orelse dim v2 <> 3
  then raise wronglength
  else linit[get(v1,2)*get(v2,3) - get(v2,2)*get(v1,3),
    get(v2,1)*get(v1,3) - get(v1,1)*get(v2,3),
    get(v1,1)*get(v2,2) - get(v2,1)*get(v1,2)];;

(* (e) Foldvect *)

fun foldrvect_emb i f e v = if dim v = 0
  then e
  else if i = 1
  then f (get(v,i), e)
  else foldrvect_emb (i-1) f (f (get(v,i), e)) v;;
fun foldrvect f e v = foldrvect_emb (dim v) f e v;;

(* (f) Quantoren für Reihungen *)

fun existsvect f = foldlvect (fn (a,b) => b orelse f(a)) false;;
fun allvect f = foldlvect (fn (a,b) => b andalso f(a)) true;;

```

Aufgabe 9-4

Geschmückte Bäume

(6 Punkte)

Folgende SML-Datei enthält die Lösung zu allen Teilaufgaben. Wir importieren zusätzlich noch `Faltung` und `map` für Listen von oben.

```

use "RS-Bintree.sml";
use "fold-map-bintree.sml";

datatype dekor = Stern | Kerze of bool | Kugel of int;

val d1 = Stern; val d2 = Kerze (true); val d3 = Kugel(4); val d4 = Kugel(12);

```

```
val b1 = build(d2, empty, empty);
val b2 = build (d3, empty, empty);
val b3 = build(d1, b1, b2);
val b4 = build (d4, b2, b3);
```

(* (a) Brennend, Geloescht *)

```
val brennend = allbintree (fn (Kerze(false)) => false | _ => true);
val geloescht = allbintree (fn (Kerze(true)) => false | _ => true);
```

(* (b) Maxkug *)

```
fun max2 (x, y) = if (x < y) then y else x;
fun max3(x, y, z) = max2(max2(x, y), z);
```

```
val maxkug = folds (fn (x, (Kugel(n)), z) => max3(x, n, z) | (x, _,
z) => max3(x, 0, z)) 0;
```

(* (c), kugelstern *)

```
val kugelstern = mapbintree (fn (Kugel(n)) => Stern | d => d);
```

(* (d), blattanz *)

```
fun blattanz (b: dekor bintree) = folds (fn (x, y, z) => x + z) 1 b;
```

(* (e), anzkug *)

```
val anzkug = folds (fn (x, (Kugel(_)), z) => x + z + 1 | (x, _, z)
=> x + z) 0;
```

(* (f), kosten *)

```
fun preis(Kugel(n)) = 2 * n | preis(Kerze(_)) = 2 | preis(Stern) = 10;
val kosten = folds (fn (x, y, z) => x + (preis y) + z) 0;
```