Ludwig-Maximilians-Universität München WS 2005/2006
Institut für Informatik Blatt 9
Prof. Dr. F. Kröger, Dr. M. Hölzl, F. Hacklinger

Übungen zu Informatik I (Lösungsvorschlag)

Aufgabe 9-1 **Pfade in Bäumen** (keine Abgabe)

```
use "bintree.sml";

datatype step = Left | Right;
type path = step list;

exception No_Node_For_Path;

fun target_node b nil = b
  | target_node b (lr::p)
    = if isemptybt b
      then raise No_Node_For_Path
      else if lr = Left
      then target_node (left b) p
      else target_node (right b) p;

val b1 = build(1, emptybt, emptybt);
val b2 = build(2, emptybt, emptybt);
val b3 = build(3, b1, b2);
val b4 = build(4, emptybt, emptybt);
val b5 = build(5, b3, b4);
```

Aufgabe 9-2 **Breitensuche für Bäume** (keine Abgabe)

```
use "bintree.sml";

fun breadth_first_emb (nil, result) = result
  | breadth_first_emb (tree_list, result) =
    let val current_tree = hd(tree_list) in
        if isemptybt(current_tree) then
            breadth_first_emb(tl(tree_list), result)
        else breadth_first_emb(tl(tree_list) @ [left(current_tree)]
             @ [right(current_tree)], result @ [root(current_tree)])
    end;

fun breadth_first_lin(tree) =
    breadth_first_emb([tree], []);
```

Aufgabe 9-3 **Funktionen für Bäume** (6 Punkte)

```
(* Aufgabe 9-3 *)

use "bintree.sml";

(* Teilaufgabe a
   Geben Sie eine SML-Funktion istblatt vom Typ a' bintree->bool an,
   die bestimmt, ob der Eingabebaum ein Blatt ist.
 *)

fun istblatt tree =
    not(isemptybt tree) andalso isemptybt(left tree) andalso isemptybt(right tree);

(* Hinweis: Den leeren Baum nicht vergessen! *)

(* Teilaufgabe b
   Geben Sie eine SML-Funktion blattanz vom Typ 'a bintree -> int an,
   welche die Anzahl der Blätter des Eingabebaumes bestimmt.
 *)

fun blattanz tree =
    if isemptybt tree then 0
    else if istblatt tree then 1
         else blattanz(left tree) + blattanz(right tree);

(* Teilaufgabe c
   Geben Sie eine SML-Funktion nichtblattanz0 vom Typ int bintree -> int an,
   welche die Anzahl der Knoten k des Eigabebaumes bestimmt, die keine Blätter sind
   und für die gilt root(k)>0
 *)

fun nichtblattanz0 tree =
    if isemptybt tree then 0
    else if istblatt tree then 0
         else (if root tree > 0 then 1 else 0) +
         nichtblattanz0(left tree) + nichtblattanz0(right tree);

(* Teilaufgabe d
   Geben Sie eine SML-Funktion allbintree vom Typ ('a -> bool) -> 'a bintree -> bool an,
   die zu einer Funktion p diejenige Funktion bestimmt,
   die entscheidet, ob p(x) für alle Knoten x eines Binärbaumes zutrifft.
 *)

fun allbintree p tree =
    isemptybt tree orelse p(root tree) andalso allbintree p (left tree)
   andalso allbintree p (right tree);

(* Hinweis:
   Auf den Typ der Funktion achten!
```

```
      fun allbintree(p,tree) = ...

   wäre vom Typ ('a -> bool) * 'a bintree -> bool.
 *)

(* zum Testen: *)
val leaf = build(1, emptybt, emptybt);

(*******************
 *                 *
 *       1         *
 *                 *
 *******************)

val tree1 = build(~2, leaf, emptybt);

(*******************
 *                 *
 *      ~2         *
 *      /          *
 *     1           *
 *                 *
 *******************)

val tree2 = build(3, emptybt, tree1);

(*******************
 *                 *
 *       3         *
 *        \        *
 *        ~2       *
 *        /        *
 *       1         *
 *                 *
 *******************)

val tree = build(1, build(2, emptybt, build(3, emptybt, emptybt)),
                 build(4, build(6, emptybt, emptybt), build(1, emptybt, emptybt)));

(*******************
 *                 *
 *       1         *
 *      / \        *
 *     /   \       *
 *    2     4      *
 *     \   / \     *
 *      3 6   1    *
 *                 *
 *******************)
```

```
fun p x = x > 0;

val test_istblatt = not(istblatt emptybt) andalso istblatt leaf
      andalso not(istblatt tree1) andalso not(istblatt tree2)
      andalso not(istblatt tree);

val test_blattanz = blattanz emptybt = 0 andalso blattanz leaf = 1
      andalso blattanz tree1 = 1 andalso blattanz tree2 = 1
      andalso blattanz tree = 3;

val test_nichtblattanz0 = nichtblattanz0 emptybt = 0
      andalso nichtblattanz0 leaf = 0 andalso nichtblattanz0 tree1 = 0
      andalso nichtblattanz0 tree2 = 1 andalso nichtblattanz0 tree = 3;

val test_allbintree = allbintree p emptybt andalso allbintree p leaf
      andalso not(allbintree p tree1) andalso not(allbintree p tree2)
      andalso allbintree p tree;
```

Aufgabe 9-4 **Geschmückte Bäume** (6 Punkte)

Folgende SML-Datei enthält die Lösung zu allen Teilaufgaben.

```
(* Aufgabe 9-4
   Gegeben sei datatype dekor=Stern | Kogel of int | Kerze of bool.
   Hierbei steht Kerze(true) für eine brennende Kerze,
   Kerze(false) für eine Kerze, die nicht brennt
   und Kugel(n) für eine Kugel der Größe n.
 *)

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

(* Teilaufgabe a
   Geben Sie zwei SML-Funktionen brennend und geloescht vom Typ dekor
   bintree -> bool an, die bestimmen, ob alle Kerzen des Eingabebaums
   brennen, bzw. ob alle Kerzen des Eingabebaums gelöscht sind.
 *)

fun brennt(Kerze false) = false
 |  brennt _ = true;

fun brennt_nicht(Kerze true) = false
 |  brennt_nicht _ = true;

(* Hinweis:

      fun brennt_nicht dekor = not (brennt dekor);
```

```
   wäre FALSCH, da für Sterne und Kugeln sowohl die Funktion brennt,
   als auch die Funktion brennt_nicht den Wert true zurückliefern müssen.
 *)

val brennend = allbintree brennt;
val geloescht = allbintree brennt_nicht;

(* Teilaufgabe b
   Geben Sie eine SML-Funktion maxkug vom Typ dekor bintree -> int an,
   welche die Größe der größten Kugel des Eingabebaums bestimmt.
   Hängen keine Kugeln am Eingabebaum, so soll maxkug den Wert 0 zurückliefern.
 *)

fun value(Kugel n) = n
 |  value _ = 0;

fun max(x,y) =
    if x<y then y
    else x;

fun maxkug tree =
    if isemptybt tree then 0
    else max(value(root tree), max(maxkug(left tree), maxkug(right tree)));

(* (c), kugelstern *)
fun change (Kugel(n)) = Stern
  | change x = x;

fun kugelstern tree =
    if isemptybt(tree) then emptybt
    else build(change(root(tree)), kugelstern(left(tree)), kugelstern(right(tree)))

(* oder Lösung mit mapbintree *)
fun mapbintree(f:'a -> 'a, tree) =
    if isemptybt(tree) then emptybt
    else build(f(root(tree)), mapbintree(f, left(tree)), mapbintree(f, right(tree)));

fun kugelstern_map tree =
    mapbintree(change, tree);

(* (d), anzkug *)
fun anz_kugel (Kugel(n)) = 1
  | anz_kugel _ = 0;

fun anzkug tree =
    if isemptybt(tree) then 0
    else anz_kugel(root(tree)) + anzkug(left(tree)) + anzkug(right(tree));


(* (e), kosten *)
```

```
fun wert_von (Kugel(n)) = 2 * n
  | wert_von Stern = 10
  | wert_von (Kerze(_)) = 5;

fun kosten tree =
    if isemptybt(tree) then 0
    else wert_von(root(tree)) + kosten(left(tree)) + kosten(right(tree));


(* zum Testen von A9-4: *)

val d1 = Stern;
val d2 = Kerze false;
val d3 = Kerze true;
val d4 = Kugel(1);
val d5 = Kugel(10);
val b1 = build(d1,emptybt,emptybt);
val b2 = build(d4,emptybt,emptybt);
val b3 = build(d5,emptybt,emptybt);
val b4 = build(d2,b1,emptybt);
val b5 = build(d3,emptybt,b2);
val b6 = build(d3,b3,emptybt);
val b7 = build(d5,b4,b5);
val b8 = build(d4,b6,b4);

val test_brennend = brennend emptybt andalso brennend b3 andalso not(brennend b4)
                    andalso brennend b5 andalso not(brennend b7);

val test_geloescht = geloescht emptybt andalso geloescht b3 andalso geloescht b4
                    andalso not(geloescht b5) andalso not(geloescht b7);

val test_maxkug = maxkug emptybt= 0 andalso maxkug b1 = 0 andalso maxkug b2 = 1
                  andalso maxkug b6 = 10 andalso maxkug b7 = 10 andalso maxkug b8 = 10;

val test_kugelstern0 = kugelstern emptybt;
val test_kugelstern'0 = kugelstern' emptybt;
val test_kugelstern1 = kugelstern b1;
val test_kugelstern'1 = kugelstern' b1;
val test_kugelstern2 = kugelstern b2;
val test_kugelstern'2 = kugelstern' b2;
val test_kugelstern5 = kugelstern b5;
val test_kugelstern'5 = kugelstern' b5;
val test_kugelstern6 = kugelstern b6;
val test_kugelstern'6 = kugelstern' b6;

val test_anzkug = anzkug emptybt = 0 andalso anzkug b1 = 0
  andalso anzkug b2 = 1 andalso anzkug b5 = 1
                  andalso anzkug b6 = 1 andalso anzkug b8 = 2;
```

```
val test_kosten = kosten emptybt = 0 andalso kosten b1 = 10
  andalso kosten b4 = 15 andalso kosten b5 = 7
                 andalso kosten b6 = 25 andalso kosten b8 = 42;
```