

## Übungen zu Informatik I

### Aufgabe 10-1

### Türme von Hanoi

(keine Abgabe)

```
exception identity

datatype Tower = One | Two | Three;

fun third(One,Two) = Three |
  third(One,Three) = Two |
  third(Two,One) = Three |
  third(Two,Three) = One |
  third(Three,One) = Two |
  third(Three,Two) = One |
  third(_,_) = raise identity;

fun toString(One) = "1" |
  toString(Two) = "2" |
  toString(Three) = "3";

fun move2string(n,a: Tower,b: Tower) =
  "Bringe Scheibe "^ Int.toString(n)^
  " von Stapel "^ (toString a)^
  " nach Stapel "^ (toString b)^ ".\n";
(* Hier sind die Typen links noetig, da evtl. mehrere toString
   im Kontext vorhanden sind. *)

fun hanoiString(0,a,b) = "" |
  hanoiString(n,a,b) = hanoiString(n-1,a,third(a,b))^
    move2string(n,a,b)^
    hanoiString (n-1,third(a,b),b);

fun hanoi(n,a,b) = print (hanoiString(n,a,b));
```

### Aufgabe 10-2

### Kryptographie

(keine Abgabe)

```
val key1 = explode "defghijklmnopqrstuvwxyzabc";;
val key2 = explode "hijklmnopqrstuvwxyzabcdefg";;

(* Hilfsfunktionen *)

exception not_found of char;;

fun index_of (xs, x) =
  let fun find_index i nil = raise (not_found x)
      | find_index i (y::ys) = if x = y
```

```

                                then i
                                else find_index (i+1) ys;
in find_index 0 xs end;;

fun index_char index = chr(index + ord("#a"));
fun dc (key,c) = if #"a" <= c andalso c <= #"z"
                then index_char(index_of (key,c))
                else c;;

(* (a) Encode Char *)

fun encode_char (key,char) = if #"a" <= char andalso char <= #"z"
                             then let val index = ord(char) - ord("#a")
                                    in List.nth(key,index) end
                             else char;;

(* (b) Encode *)

fun encode (key,string) =
  let fun enc [] = []
        | enc (c::cs) = (encode_char(key,c))::(enc cs)
      in implode (enc (explode string)) end;;

(* (c) Decode *)

fun decode (k,s) =
  let fun dec [] = []
        | dec (c::cs) = (dc(k,c))::(dec cs)
      in implode (dec (explode s)) end;;

(* (d) Frohe Weihnachten ... *)

(* (e) Encode2 *)

fun encode2 (k1,k2,s) =
  let val ec = encode_char
        fun enc [] = []
              | enc [c] = [ec(k1,c)]
              | enc (c1::c2::cs) = (ec(k1,c1))::(ec(k2,c2))::(enc cs)
      in implode (enc (explode s)) end;;

(* (f) Decode2 *)

fun decode2 (k1,k2,s) =
  let fun dec [] = []
        | dec [c] = [dc(k1,c)]
        | dec (c1::c2::cs) = (dc(k1,c1))::(dc(k2,c2))::(dec cs)
      in implode (dec (explode s)) end;;

(* (g) ... und ein gutes neues Jahr *)

```

**Aufgabe 10-3**

**Weihnachtsbaum**

(keine Abgabe)

```

(* Xmas tree *)

fun ntimes(s,0) = "" |
  ntimes(s,n) = s ^ ntimes(s,n-1);

fun alternatentimes(s,t,0) = "" |
  alternatentimes(s,t,n) = s ^ alternatentimes(t,s,n-1);

fun inner1(k) = alternatentimes(".",",",2*k+1)

fun inner2(k) = alternatentimes(",",".",2*k+3)

fun inner1X(k) = alternatentimes("#","o",2*k+1)

fun inner2X(k) = alternatentimes("o","#",2*k+3)

fun level1 inner (n,k) =
  ntimes(" ",n-k) ^ "/" ^ inner(k) ^ "\\\"
  ^ ntimes(" ",n-k) ^ "\n";

fun level2 inner (n,k) =
  ntimes(" ",n-k-1) ^ "/" ^ inner(k) ^ "\\\"
  ^ ntimes(" ",n-k-1) ^ "\n";

fun level i1 i2 (n,k) = if n>k then
  level1 i1 (n,k)^level2 i2 (n,k)^level i1 i2 (n,k+1)
  else "";

fun xmas i1 i2 (n) =
  ntimes(" ",n+1)^"*"^ntimes(" ",n+1)^\n"^level i1 i2 (n,0)^
  ntimes("^",n)^[_] ^ntimes("^",n)^\n\n\n";

fun printX (n) = print (xmas inner1X inner2X n);

```

#### Aufgabe 10-4

#### Listenfunktionen

(4 Punkte)

(\* (a) Listeq \*)

```

fun listeq (nil,nil) = true
  | listeq ((x::xs),(y::ys)) = x = y andalso listeq (xs,ys)
  | listeq (_,_) = false;;

```

(\* (b) Listeq1 \*)

```

fun listeq1 (p,nil,nil) = true
  | listeq1 (p,(x::xs),(y::ys)) = p(x,y) andalso listeq1 (p,xs,ys)
  | listeq1 (_,_,_) = false;;

```

(\* (c) Ordered \*)

```

fun ordered (p,nil) = true

```

```
| ordered (p, [x]) = true
| ordered (p, (x1::x2::xs)) = p(x1,x2) andalso ordered (p, (x2::xs));;
```

(\* (d) Swap2 \*)

```
fun swap2 [] = []
  | swap2 [x] = [x]
  | swap2 (x1::x2::xs) = x2::x1::(swap2 xs);;
```

### Aufgabe 10-5

### „Bounty“ (Josephus-Problem)

(8 Punkte)

(\* iter n f bestimmt die n-fache Komposition einer funktion mit sich selbst; dh. iter n f = f o ... o f (n mal). \*)

```
fun iter 0 f = (fn x => x) | iter n f = f o (iter (n-1) f);
```

(\* rob nimmt das erste element einer Liste und haengt es ans Ende \*)

```
fun rob nil = nil | rob (x::xs) = xs @ [ x ];
```

(\* once d loescht das d-te Element einer Liste; die ersten d-1 werden nach hinten geschoben \*)

```
fun once d = tl o (iter (d-1) rob);
```

(\* josephus k d l liefert die Liste der Personen, die beim Auszaehlen von n Personen aus der Liste l mit Intervallschritt d uebrigbleiben. \*)

```
fun josephus k d = iter k (once d);
```

(\* lessthan n ist die Liste aller nichtnegativen ganzen Zahlen < n \*)

```
fun lessthan 0 = nil | lessthan n = lessthan (n-1) @ [ n-1];
```

(\* bounty bekommt man nun durch Kombination von lessthan und josephus; beachte, das die Verwendung von Tupel hoehere Typen vermeidet \*)

```
fun bounty (n, k, d) = josephus k d (lessthan n);
```