Ludwig-Maximilians-Universität München
Institut für Informatik
Prof. Dr. F. Kröger, Dr. M. Hölzl, F. Hacklinger

WS 2005/2006
Probeklausur

Probeklausur Informatik I (Lösungsvorschlag)

Aufgabe 1 **Verständnisfragen** (2+1+2 Punkte)

a) 1. Fall: x = 0. Der Aufruf von terminiert. 2. Fall: $x > 0$. Der Aufruf von f führt dann zur Auswertung des `else`-Zweiges des zweiten `if`-Konstrukts. f wird rekursiv mit $f(x-1)$ aufgerufen und solange $x > 0$ immer wieder bis $x = 0$. Damit terminiert f für Aufrufe mit $x > 0$. 3. Fall: $x < 0$: Der `if`-Zweig des zweiten `if`-Konstrukts wird gewählt. Es wird rekursiv $f(-x+1)$ aufgerufen. Wegen $x < 0$ gilt: der Parameter des resultierenden Aufrufs ist positiv. Für Aufrufe mit $x > 0$ terminiert f.

b) Der Fehler liegt im `let`-Konstrukt: Wird in dem Konstrukt ein Wert an einen Namen gebunden, muss vor dem Namen ein `val` stehen.

c) Da niemals eine monomorphe Operation auf ein Element der Liste angewendet wird, spielt der Typ der Element keine Rolle; die Funktion ist daher polymorph.

Aufgabe 2 **SML** (1+2+2+3+4 Punkte)

```
use "bintree.sml";

fun fromTo(m,n) = if m > n then []
  else (m::(fromTo(m+1,n)));;

fun zip nil _ = []
  | zip _ nil = []
  | zip (x::xs) (y::ys) = (x,y)::(zip xs ys);;

fun listpos l = zip (fromTo(1,length(l))) l;;

fun listpos' l =
    let fun iter nil _ = []
  | iter (x::xs) n = (n,x)::(iter xs (n+1))
    in iter l 1 end;;

fun prefix(s,l) = map (fn x => s^x) l;;

exception emptylist;;

fun taken _ 0 = []
  | taken nil _ = raise emptylist
  | taken (x::xs) n = x::(taken xs (n-1));;

exception wronglength;;
```

```
fun take3 l = if (length l) mod 3 = 0
      then taken l ((length l) div 3)
      else raise wronglength;;

fun tfind(p,tree) =
    if isemptybt(tree) then []
    else if p(root(tree)) then [root(tree)] @ tfind(p, left(tree))
       @ tfind(p, right(tree))
         else tfind(p, left(tree)) @ tfind(p right(tree));;
```

Aufgabe 3 **Datatype-Deklaration** (3+3 Punkte)

```
datatype 'a twoThree = nothing
     | two of 'a * 'a twoThree * 'a twoThree
     | three of 'a * 'a twoThree * 'a twoThree * 'a twoThree;;

val t1 = two(1,
     three(2,
   two(3,nothing,nothing),
   nothing,
   three(4,nothing,nothing,nothing)),
     two(5,
 nothing,
 nothing));;

fun isNothing nothing = true
  | isNothing _ = false;;

fun flatten nothing = []
  | flatten (two(x,t1,t2))
      = (x::((flatten t1) @ (flatten t2)))
  | flatten (three(x,t1,t2,t3))
      = (x::((flatten t1) @ (flatten t2) @ (flatten t3)));;

fun count2 nothing = 0
  | count2 (two(_,t1,t2)) = 1 + count2(t1) + count2(t2)
  | count2 (three(_,t1,t2,t3)) = count2(t1) + count2(t2) + count2(t3);;

fun count3 nothing = 0
  | count3 (two(_,t1,t2)) = count3(t1) + count3(t2)
  | count3 (three(_,t1,t2,t3)) = 1 + count3(t1) + count3(t2) + count3(t3);;

fun count23 t = (count2 t, count3 t);;
```

Aufgabe 4 **Bäume** (5 Punkte)

```
datatype 'a ntree = n_empty | n_build of 'a * ('a ntree) list

fun any f nil = false
```

```
  | any f (x::xs) = f x orelse any f xs;;

fun contains(x,n_empty) = false
  | contains(x,n_build(y,ts)) = x = y orelse any (fn t => contains(x,t)) ts;;
```

Aufgabe 5 **Terminierung** (2+4 Punkte)

a) Die Auswertung wurde in vereinfachter „Kurzschreibweise“(siehe Kap. 2 im Skript) notiert.

$$\begin{aligned} f(31,-28) &= 1 + f(33,-31) \\ &= 1 + (1 + f(35,-34)) \\ &= 1 + (1 + (1 + f(37,-37))) \\ &= 1 + (1 + (1 + (1 + f(39,-40)))) \\ &= 1 + (1 + (1 + (1 + -1))) \\ &= 3 \end{aligned}$$

b) Seien $A = \mathbb{Z} \times \mathbb{Z}$, $M = \mathbb{N}$, $\prec = <$, $h : A \to M$ mit

$$h(x,y) = \begin{cases} x + y & \text{falls } x + y \geq 0, \\ 0 & \text{sonst.} \end{cases}$$

1. Fall: $x + y < 0$. In diesem Fall finden keine rekursiven Aufrufe statt und die Funktion f terminiert sofort.

2. Fall: $x + y \geq 0$. In diesem Fall findet ein rekursiver Aufruf $f(x+2, y-3)$ statt. Es ist zu zeigen, dass gilt.

1. $(x + 2, y - 3) \in \mathbb{Z} \times \mathbb{Z}$
2. $h(x + 2, y - 3) < h(x, y)$

Offensichtlich ist für $y \in \mathbb{Z}$ auch $x + 2 \in \mathbb{Z}$ und $y - 3 \in \mathbb{Z}$ und für $(x + 2, y - 3)$ gilt:

$$\begin{aligned} h(x+2, y-3) &= x + 2 + (y - 3) \\ &= x + y - 1 \\ &< x + y \\ &= h(x,y) \end{aligned}$$

Aufgabe 6 **BNF-Grammatiken** (5 Punkte)

a) Die folgende Grammatik leistet das Gewünschte:

$$\begin{aligned} \langle \mathit{LowerChar} \rangle &= \texttt{a} \mid \ldots \mid \texttt{z} \\ \langle \mathit{AlphaChar} \rangle &= \texttt{A} \mid \ldots \mid \texttt{Z} \mid \langle \mathit{LowerChar} \rangle \\ \langle \mathit{AnyChar} \rangle &= \langle \mathit{AlphaChar} \rangle \mid \_ \\ \langle S \rangle &= \langle \mathit{AlphaChar} \rangle \ \{\langle \mathit{AnyChar} \rangle\}^* \mid \_ \ \langle \mathit{LowerChar} \rangle \ \{\langle \mathit{AnyChar} \rangle\}^* \end{aligned}$$

b) Mögliche Ableitungen sind:

$$
\begin{aligned}
\langle S\rangle &\rightarrow \langle \mathit{AlphaChar}\rangle\ \{\langle \mathit{AnyChar}\rangle\}^*\\
&\rightarrow \langle \mathit{AlphaChar}\rangle\ \langle \mathit{AnyChar}\rangle\ \langle \mathit{AnyChar}\rangle\ \langle \mathit{AnyChar}\rangle\ \langle \mathit{AnyChar}\rangle\ \langle \mathit{AnyChar}\rangle\\
&\rightarrow^* \langle \mathit{LowerChar}\rangle\ \langle \mathit{AlphaChar}\rangle\ \langle \mathit{AlphaChar}\rangle\ \langle \mathit{AlphaChar}\rangle\ \langle \mathit{AlphaChar}\rangle\ \langle \mathit{AlphaChar}\rangle\\
&\rightarrow^* \langle \mathit{LowerChar}\rangle\ \langle \mathit{LowerChar}\rangle\ \langle \mathit{LowerChar}\rangle\ \langle \mathit{LowerChar}\rangle\ \langle \mathit{LowerChar}\rangle\ \langle \mathit{LowerChar}\rangle\\
&\rightarrow^* \texttt{printf}
\end{aligned}
$$

und

$$
\begin{aligned}
\langle S\rangle &\rightarrow \_\ \langle \mathit{LowerChar}\rangle\ \{\langle \mathit{AnyChar}\rangle\}^*\\
&\rightarrow \_\ \langle \mathit{LowerChar}\rangle\\
&\rightarrow \_\texttt{x}
\end{aligned}
$$