Lösungen der ALP1-Nachklausur
9. April 2001

1. **Aufgabe (5-10 Min.)** *4 Punkte*
   Schreiben Sie eine Haskell-Funktion $absort$, die testet, ob eine gegebene Liste von Zahlen absteigend sortiert ist.

   Lösung:

$$
\begin{array}{ll}
absort :: \; (Num\ a,\ Ord\ a) \; => \; [a] \; \rightarrow \; Bool & \\
absort \; [] & = \; True \\
absort \; [x] & = \; True \\
absort \; (x : ls) & \mid x >= head\ ls \;\; = \; absort\ ls \\
 & \mid otherwise \quad\;\; = \; False
\end{array}
$$

2. **Aufgabe (20 Min.)** *8 Punkte*

   (a) Schreiben Sie eine rekursive Haskell-Funktion $paare$, die zu einer Liste die Liste aller möglichen Paare erzeugt, z.B. $paare\ [3,5,2] \ = \ [(3,5),(3,2),(5,2)]$ oder eine Permutation davon.

   (b) Wenden Sie Ihre Funktion auf das Beispiel aus (a) an und reduzieren sie diesen Ausdruck schrittweise auf seinen Wert.

   Lösung:

$$
\begin{array}{ll}
(a)\ paare :: \; [a] \; \rightarrow \; [(a,a)] & \\
paare \; [] & = \; [] \\
paare \; (x : xs) & = \; map \; (f \;\; x) \;\; xs \; ++ \, paare \;\; xs \\
f \; :: \; a \; \rightarrow \; b \; \rightarrow (a,b) & \\
f \;\; x \;\; y & = \; (x,y)
\end{array}
$$

$$
\begin{array}{l}
(b)\ paare \;\; [3,5,2] \; \rightarrow \; map \;\; (f \;\; 3) \;\; [5,2] \;\; ++ \, paare \;\; [5,2] \\
\quad \rightarrow \; (f \;\; 3 \;\; 5) : map \;\; (f \;\; 3) \;\; [2] \; ++ \, paare \;\; [5,2] \\
\quad \rightarrow \; (3,5) : (f \;\; 3 \;\; 2) : map \;\; (f \;\; 3) \;\; [] \;\; ++ \, paare \;\; [5,2] \\
\quad \rightarrow \; (3,5) : (3,2) : map \;\; (f \;\; 3) \;\; [] \;\; ++ \, paare \;\; [5,2] \\
\quad \rightarrow \; (3,5) : (3,2) : [] \;\; ++ \, paare \;\; [5,2] \\
\quad \rightarrow \; [(3,5),(3,2)] \;\; ++ \, paare \;\; [5,2] \\
\quad \rightarrow \; [(3,5),(3,2)] \;\; ++ \, map \;\; (f \;\; 5) \;\; [2] \;\; ++ \, paare \;\; [2] \\
\quad \rightarrow \; [(3,5),(3,2)] \;\; ++ \, (f \;\; 5 \;\; 2) : map \;\; (f \;\; 5) \;\; [] \;\; ++ \, paare \;\; [2] \\
\quad \rightarrow \; [(3,5),(3,2)] \;\; ++ \, (5,2) : [] \;\; ++ \, paare \;\; [2] \\
\quad \rightarrow \; [(3,5),(3,2)] \;\; ++ \, (5,2) : [] \;\; ++ \, map \;\; (f \;\; 2) \;\; [] \;\; ++ \, paare \;\; [] \\
\quad \rightarrow \; [(3,5),(3,2)] \;\; ++ \, [(5,2)] \;\; ++ \, [] \;\; ++ \, paare \;\; [] \\
\quad \rightarrow \; [(3,5),(3,2)] \;\; ++ \, [(5,2)] \;\; ++ \, [] \;\; ++ \, [] \\
\quad \rightarrow \; [(3,5),(3,2),(5,2)] \;\; ++ \, [] \;\; ++ \, [] \\
\quad \rightarrow \; [(3,5),(3,2),(5,2)] \;\; ++ \, [] \\
\quad \rightarrow \; [(3,5),(3,2),(5,2)]
\end{array}
$$

3. **Aufgabe (15 Min.)** *6 Punkte*

Schreiben Sie je eine Haskell-Funktion ((a) $poseven$, (b) $counteven$, (c) $tripeleven$), die zu einer gegebenen Liste von ganzen Zahlen die folgende Frage beantwortet:

(a) Auf welcher Listenposition steht zum ersten Mal (von links nach rechts betrachtet) eine gerade Zahl?

(b) Wieviele gerade Zahlen gibt es in der Liste?

(c) Gibt es drei aufeinander folgende, gerade Zahlen in der Liste?

Lösung:

$(a)\ poseven ::\ [Int]\ \longrightarrow Int$

$poseven\ []\quad = if\ poss\ ==\ []\ then\ error\ "Die\ Liste\ enthält\ keine\ gerade\ Zahl."$
$\quad else\ head\ poss$
$\quad where\ poss\ =\ [i \mid (x,i) \leftarrow zip\ xs\ [0\cdot\cdot],\ even\ x]$

$(b)\ counteven ::\ [Int]\ \longrightarrow Int$

$counteven\ xs\quad = length\ (filter\ even\ xs)$

$(c)\ tripeleven ::\ [Int]\ \longrightarrow Bool$

$tripeleven\ xs\quad \mid length\ xs\ <\ 3\ =\ False$
$\quad \mid even\ (xs\ !!\ 0)\ \&\&\ even\ (xs\ !!\ 1)\ \&\&\ even\ (xs\ !!\ 2)\ =\ True$
$\quad \mid otherwise\ =\ tripeleven\ (tail\ xs)$

4. **Aufgabe (20 Min.)** *6 Punkte*

Ein Bild sei als eine Folge von Zeilen gleicher Länge dargestellt. Schreiben Sie eine Haskell-Funktion $linksrot$, die ein Bild um $90°$ nach links dreht.

Lösung:

$sp ::\ [String]\ \rightarrow\ IO()$

$sp\quad = putStr.sptransponiere$

$sptransponiere ::\ [String]\ \rightarrow\ String$

$sptransponiere\ bild\quad =\ foldr\ (++)\ []\ (foldr\ op\ []\ (map\ reverse\ bild))$
$\quad where$
$\quad op\ string\ []\ =\ [[ch]\ ++\ "\backslash n" \mid ch\ \leftarrow\ string]$
$\quad op\ (ch : rest)\ (cha : rest')\ =\ (ch : cha) : op\ rest\ rest'$

5. **Aufgabe (15 Min.)** *4 Punkte*

Beweisen Sie durch strukturelle Induktion, dass

$$take\ (n1\ +\ n2)\ l\ =\ (take\ n1\ l) ++ take\ n2\ (drop\ n1\ l)$$

Begründen Sie jeden Ihrer Beweisschritte.

Lösung:

$Beweis\ \ddot{u}ber\ die\ Struktur\ von\ l$

$Ind.Ann.: take\ (n1 + n2)\ l = (take\ n1\ l) ++ take\ n2\ (drop\ n1\ l)$

$l = []$

$take\ (n1 + n2)\ []$

$= []$ $(Def.\ take)$

$(take\ n1\ []) ++ take\ n2\ (drop\ n1\ [])$

$= [] ++ take\ n2\ (drop\ n1\ [])$ $(Def.\ take)$

$= [] ++ take\ n2\ []$ $(Def.\ drop)$

$= [] ++ []$ $(Def.\ take)$

$= []$ $(Def.\ (++))$

$l = (a : l),\ n1 > 0$

$take\ n1\ (a : l) ++ take\ n2\ (drop\ n1\ (a : l))$

$= a : (take\ (n1 - 1)\ l) ++ take\ n2\ (drop\ (n1 - 1)\ l)$ $(Def.\ take\ und\ drop)$

$= a : (take\ (n1 - 1)\ l ++ take\ n2\ (drop\ (n1 - 1)\ l))$ $(Def.\ (++))$

$= a : (take\ (n1 - 1 + n2)\ l)$ $(Ind.\ Ann.)$

$= take\ (n1 + n2)\ (a : l)$ $(Def.\ take)$

$l = (a : l),\ n1 = 0$

$take\ 0\ (a : l) ++ take\ n2\ (drop\ 0\ (a : l))$

$= [] ++ take\ n2\ (a : l)$ $(Def.\ take\ und\ drop)$

$= take\ n2\ (a : l))$ $(Def.\ (++))$

$take\ (0 + n2)\ (a : l)$

$= take\ n2\ (a : l)$

6. **Aufgabe (20 Min.)** *6 Punkte*

(a) Geben Sie den Typ folgender Funktionen an:

$$f\ \ x = ('i','j') : x$$
$$g\ \ i\ \ j\ \ x\ = \ (i, j) : x$$
$$h\ \ i\ \ j\ \ x\ = \ (i\ \ j) : x$$

(b) Geben Sie zu jedem dieser Beispiele eine Beispielanwendung an und reduzieren Sie diese auf ihren Wert.

Lösung:

(a)

$$f :: [(Char, Char)] \longrightarrow [(Char, Char)]$$
$$g :: a \longrightarrow b \longrightarrow [(a, b)] \longrightarrow [(a, b)]$$
$$h :: (a \longrightarrow b) \longrightarrow a \longrightarrow [b] \longrightarrow [b]$$

(b)

$$f\ [('a','b')] \rightarrow \cdots \rightarrow [('i','j'), ('a','b')]$$
$$g\ 7\ "Hallo"\ []\ \rightarrow \cdots \rightarrow [(7, "Hallo")]$$
$$h\ even\ 7\ [False, True] \rightarrow \cdots \rightarrow [False, False, True]$$