Lösungen zur ALP1-Klausur.
12. Februar 2001

1. **Aufgabe (5-10 Min.)** *4 Punkte*
   Schreiben Sie (ohne Verwendung der Systemfunktionen lcm und gcd) eine Haskell-Funktion zur Berechnung des kleinsten, gemeinsamen Vielfachen (kgV) zweier natürlicher Zahlen.

   Lösung:

   $$\begin{array}{ll} kgV :: Int \rightarrow Int \rightarrow Int & \\ kgV\ n\ m & \mid m \leq 0 \mid\mid n \leq 0 = error\ \text{"}Die\ Argumente\ müssen\ positiv\ sein.\text{"} \\ & \mid otherwise\ = head\,[x \mid x \leftarrow [1\,..],\ x\ \text{`}mod\text{`}\ n\ == 0,\ x\ \text{`}mod\text{`}\ m\ == 0] \end{array}$$

2. **Aufgabe (25 Min.)** *8 Punkte*

   (a) Zu einer Menge M heißt $P(M) := \{T\,|T \subseteq M\}$ Potenzmenge von M. Zur Erinnerung: die Potenzmenge ist die Menge aller Teilmengen einer gegebenen Menge. Übertragen Sie dieses Konzept auf endliche Listen und definieren Sie einen entsprechenden Operator *pot* $::\ [t] \longrightarrow [[t]]$.
   Beispiel: pot $[1,2,3]\ = [[],[1],[2],[3],[1,2],[1,3],[2,3],[1,2,3]]$.

   Hinweis: diese Aufgabe erlaubt mehrere Lösungen, insbesondere ist die Reihenfolge der Elemente in *pot l* unerheblich.

   (b) Wenden Sie Ihre Funktion *pot* auf die Liste $[1,2]$ an und reduzieren Sie Ihren Ausdruck, dass Sie $[[],[1],[2],[1,2]]$ bzw. eine Permutation davon erhalten.

   Lösung:

   $$\begin{array}{ll} a)\ pot :: [a] \rightarrow [[a]] & \\ pot\ [] & = [[]] \\ pot\ (a : la) & = pot\ la\ ++\ map\ (a :)\ (pot\ la) \end{array}$$

   $$\begin{array}{l} b)\ pot\ [1,2] \rightarrow pot\ [2]\ ++\ map\ (:1)\ (pot\ [2]) \\ \quad \rightarrow pot\ []\ ++\ map\ (:2)\ (pot\ [])\ ++\ map\ (:1)\ (pot\ [2]) \\ \quad \rightarrow [[]]\ ++\ [[2]]\ ++\ map\ (:1)\ (pot\ []\ ++\ map\ (:2)\ (pot[])) \\ \quad \rightarrow [[],[2]]\ ++\ map\ (:1)\ ([[]]\ ++\ map\ (:2)\ (pot\ [])) \\ \quad \rightarrow [[],[2]]\ ++\ map\ (:1)\ ([[]]\ ++\ [[2]]) \\ \quad \rightarrow [[],[2]]\ ++\ map\ (:1)\ [[],[2]] \\ \quad \rightarrow [[],[2]]\ ++\ [[1],[1,2]] \\ \quad \rightarrow [[],[2],[1],[1,2]] \end{array}$$

3. **Aufgabe (10 Min.)** *6 Punkte*
   Gegeben sei eine Liste und ein Prädikat, das über Elementen dieser Liste definiert ist. Schreiben Sie je eine Haskell-Funktion, die

   (a) das erste Element liefert, das das Prädikat erfüllt

(b) alle Elemente liefert, die das Prädikat erfüllen

(c) die Liste der Indizes liefert, auf denen Elemente stehen, die das Prädikat erfüllen.

Lösung:

$$
\begin{array}{ll}
a)\ first :: [t] \longrightarrow (t \longrightarrow Bool) \longrightarrow t & \\
first\ ls\ p & = head\ (myall\ ls\ p) \\
b)\ myall :: [t] \longrightarrow (t \longrightarrow Bool) \longrightarrow [t] & \\
myall\ ls\ p & = [x \mid x \leftarrow ls,\ p\ x] \\
c)\ indizes :: [t] \longrightarrow (t \longrightarrow Bool) \longrightarrow [Int] & \\
indizes\ l\ p & = [i \mid (a,i) \leftarrow zip\ l\ [0\,..],\ p\ a] \\
c')\ indizes\ ls\ p & = indizesaux\ p\ ls\ 0 \\
indizesaux\ p\ []\ \_ & = [] \\
indizesaux\ p\ (x:ls)\ n & \mid p\ x = n : indizesaux\ p\ ls\ (n+1) \\
 & \mid otherwise = indizesaux\ p\ ls\ (n+1)
\end{array}
$$

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

Gegeben sei ein Text als Liste seiner Wörter. Schreiben Sie eine Haskell-Funktion sp, die solch einen Text spaltenweise ausgibt, z.B.:

$> sp\ [\text{"}das\text{"}, \text{"}ist\text{"}, \text{"}schön\text{"}]$

$$
\begin{array}{lll}
d & i & s \\
a & s & c \\
s & t & h \\
 & & ö \\
 & & n
\end{array}
$$

Lösung:

$$
\begin{array}{ll}
sp :: [String] \rightarrow IO() & \\
sp & = putStr.sptransponiere.spnormalisiere \\
spnormalisiere :: [String] \rightarrow [String] & \\
spnormalisiere\ s & = map\ (indentation\ blanks)\ s \\
 & where \\
 & indentation\ blanks\ str = str ++ [' \ ' \mid x \leftarrow [1\,..\,blanks - length\ str]] \\
 & blanks = foldl\ (max)\ 0\ (map\ length\ s) \\
sptransponiere :: [String] \rightarrow String & \\
sptransponiere\ s & = foldr\ (++)\ []\ (foldr\ op\ []\ s) \\
 & where \\
 & op\ string\ [] = [[ch] ++ \text{"}\backslash n\text{"} \mid ch \leftarrow string] \\
 & op\ (ch:rest)\ (cha:rest') = (ch:cha) : op\ rest\ rest'
\end{array}
$$

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

Beweisen Sie durch strukturelle Induktion, dass $summe\ (l1\ ++\ l2) = summe\ l1 + summe\ l2$ gilt, wobei $summe = foldr\ (+)\ 0$. Begründen Sie jeden Ihrer Beweisschritte.

Lösung:

$Beweis\ über\ die\ Struktur\ von\ l1$

$Ind.Ann.:\ summe\ (l1\ \ ++\ l2)\ =\ summe\ l1\ +\ summe\ l2$

$l1\ =\ []$

$summe\ ([]\ ++ l2)$

$=\ summe\ l2 \qquad (Def.(++))$

$summe\ []\ +\ summe\ l2$

$=\ foldr\ (+)\ 0\ []\ +\ summe\ l2 \qquad (Def.summe)$

$=\ 0\ +\ summe\ l2 \qquad (Def.foldr)$

$=\ summe\ l2$

$summe\ ((a:l1)\ ++ l2)$

$=\ foldr\ (+)\ 0\ (a:(l1\ \ ++ l2)) \qquad (Def.summe)$

$=\ a\ +\ (foldr\ (+)\ 0\ (l1\ ++ l2)) \qquad (Def.foldr)$

$=\ a\ +\ (summe\ (l1\ ++ l2)) \qquad (Def.summe)$

$=\ a\ +\ (summe\ l1\ +\ summe\ l2) \qquad (Ind.\ Ann.)$

$=\ (a\ +\ summe\ l1)\ +\ summe\ l2 \qquad (Assoziativ.\ von\ +)$

$=\ foldr\ (+)\ 0\ (a:l1)\ +\ summe\ l2 \qquad (Def.summe\ und\ Def.\ von\ foldr)$

$=\ summe\ (a:l1)\ +\ summe\ l2 \qquad (Def.summe)$

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

Gegeben seien die Funktionen:

$$run :: [t] \longrightarrow u \longrightarrow u$$

und

$$execute :: t \longrightarrow u \longrightarrow u$$

mit

$$\begin{aligned} run\ []\ s &= s \\ run\ (i\ :\ il)\ s &= run\ il\ (execute\ i\ s) \end{aligned}$$

und $execute$ beliebig. Beweisen Sie durch strukturelle Induktion, dass

$$run\ (il1\ \text{++}\ il2)\ s\ =\ run\ il2\ (run\ il1\ s)$$

für alle Argumente s und $il2$ sowie endliche Listen $il1$ gilt. Begründen Sie jeden Ihrer Beweisschritte.

Lösung:

$Beweis\ über\ die\ Struktur\ von\ il1$

$Ind.Ann.\ run\ (il1\ \ ++\ il2)\ s\ =\ run\ il2\ (run\ il1\ s)$

$il1\ =\ []$

$run\ ([]\ ++ il2)\ s$

$=\ run\ il2\ s \qquad (Def.\ ++)$

$run\ il2\ (run\ []\ s)$

$=\ run\ il2\ s \qquad (Def.run)$

$il1\ =\ a:il1$

$run\ (a:il1\ ++ il2)\ s$

$=\ run\ (a:(il1\ ++ il2))\ s \qquad (Def.\ ++)$

$=\ run\ (il1\ ++ il2)\ (execute\ a\ s) \qquad (Def.run)$

$=\ run\ il2\ (run\ il1\ (execute\ a\ s)) \qquad (Ind.\ Ann.)$

$=\ run\ il2\ (run\ (a:il1)\ s) \qquad (Def.run)$