

**6. Übungsaufgabe zu**  
**Fortgeschrittene funktionale Programmierung**  
**Thema: Erschöpfende Suche, Generator/Transformer/Filter-Prinzip,**  
**Arrays**  
**ausgegeben: Mi, 09.05.2012, fällig: Mi, 16.05.2012**

Für dieses Aufgabenblatt sollen Sie Haskell-Rechenvorschriften zur Lösung der im folgenden angegebenen Aufgabenstellungen entwickeln und für die Abgabe in einer Datei namens **AufgabeFFP6.hs** in Ihrem Gruppenverzeichnis ablegen, wie gewohnt auf oberstem Niveau. Kommentieren Sie Ihre Programme aussagekräftig und benutzen Sie, wo sinnvoll, Hilfsfunktionen und Konstanten.

Sei  $f = (f_1, \dots, f_k)$ ,  $k \geq 2$ , eine nichtleere endliche Folge ganzer Zahlen und  $g = (g_1, \dots, g_{k-1})$  eine um 1 kürzere Folge der arithmetischen Operatoren  $+$ ,  $*$ ,  $-$  und  $./.$  mit  $./.$  ganzzahlige Division mit Rest.

Eine *Auswertung* von  $f$  bzgl.  $g$  ist der Wert des Ausdrucks

$$f_1 \ g_1 \ f_2 \ g_2 \ \dots \ f_{k-1} \ g_{k-1} \ f_k$$

ohne Beachtung der Regel Punkt- vor Strichrechnung.

*Beispiele:*

$f = (3, 5, 2, (-2), 7, 0)$  hat bzgl.  $g = (+, ./., *, +, -)$  den Wert  $(3 + 5 ./ 2 * (-2) + 7 - 0) = (-1)$ ; bzgl.  $g' = (*, -, +, ./., +)$  den Wert  $(3 * 5 - 2 + (-2) ./ 7 + 0) = 1$ .

$f = (4, 2, 3, (-4), 5, 2)$  hat bzgl.  $g = (+, ./., *, +, -)$  den Wert  $(4 + 2 ./ 3 * (-4) + 5 - 2) = (-5)$ ; bzgl.  $g' = (*, -, +, ./., +)$  den Wert  $(4 * 2 - 3 + (-4) ./ 5 + 2) = 2$ .

- Schreiben Sie eine Haskell-Rechenvorschrift `eval :: Array Int Int -> Array Int (Int->Int->Int) -> Int`, die angewendet auf eine Zahlenfolge  $f$  und eine Operatorfolge  $g$  passender Länge den Wert von  $f$  bzgl.  $g$  im Sinne des vorstehend eingeführten Auswertungsbegriffs bestimmt. Divisionen durch 0 führen dabei zu einem Fehlerabbruch der Auswertung.

*Beispiele:*

```
eval (array (1,3) [(1,1),(2,2),(3,3)]) (array (1,2) [(1,(+)),(2,(-))])
->> 0
eval (array (1,3) [(1,1),(2,2),(3,3)]) (array (1,2) [(1,(*)),(2,(+))])
->> 5
eval (array (1,3) [(1,1),(2,2),(3,3)]) (array (1,2) [(1,(-)),(2,(*))])
->> (-3)
```

Die Funktion `eval` wird nur auf zueinander passende Zahlen- und Operationsfolgen angewendet.

- Die Funktion `yield :: Array Int Int -> Int -> [Array Int (Int->Int->Int)]` soll angewendet auf eine Zahlenfolge  $f$  und einen Zielwert  $w$  eine Liste derjenigen Folgen arithmetischer Operationen liefern, so dass die Auswertung der Argumentliste bzgl. dieser Operationsfolge den Zielwert  $w$  ergibt. Die Reihenfolge verschiedener Operationsfolgen innerhalb der Ergebnisliste ist dabei unerheblich.

*Beispiele:*

```
yield array (1,3) [(1,1),(2,2),(3,3)] 6
->> [array (1,2) [(1,(+)),(2,(+))],array (1,2) [(1,(*)),(2,(*))]]
yield array (1,3) [(1,1),(2,2),(3,3)] 4
->> []
yield array (1,3) [(1,1),(2,2),(3,3)] 0
->> [array [(1,2) (1,(+)),(2,(-))],array (1,2) [(1,div),(2,(*))],
      array (1,2) [(1,(*)),(2,div)],array (1,2) [(1,div),(2,div)]]
```

Implementieren Sie zwei unterschiedliche Varianten

```
- yield_bt :: Array Int Int -> Int -> [Array Int (Int->Int->Int)]
- yield_gtf :: Array Int Int -> Int -> [Array Int (Int->Int->Int)]
```

die funktional äquivalent zur Funktion `yield` sind, wobei `yield_bt` sich auf das Backtracking-Funktional aus Kapitel 3.2 aus der Vorlesung und `yield_gtf` sich auf das generate/transform/filter-Prinzip abstützt.

Implementieren Sie für `yield_gtf` drei Funktionen `generate`, `transform` und `filt`, so dass sich `yield_gtf` als sequentielle Komposition `filt . transform . generate`

ergibt. Die Funktion `generate` erzeugt alle Operationsfolgen (passender Länge), `transform` übernimmt die Auswertung, `filt` wählt die Auswertungen mit passendem Zielwert aus.

(Hinweis: Eine unmittelbare Ausgabe der Resultate ist nicht möglich, da der Resultattyp von `yield_bt` bzw. `yield_gtf` nicht in der Klasse `Show` liegt. Beide Funktionen werden nur mit mindestens zweielementigen Zahlenfolgen aufgerufen.)

- Machen Sie den Datentyp `Array Int (Int->Int->Int)` zu einer Instanz der Klasse `Show`. Die arithmetischen Operationen aus den vorigen Aufgaben sollen als Zeichenreihen

```
"plus", "minus", "times", "div"
```

dargestellt werden. Für mögliche andere Operationen wird keine besondere Ausgabe verlangt.

*Hinweis:* Verwenden Sie für diese Teilaufgabe den Aufruf `hugs -98 +o` oder `runhugs -X-98 +o` und ergänzen Sie `{-# LANGUAGE FlexibleInstances #-}` im Quelltext.