

**5. Übungsaufgabe zu  
Fortgeschrittene funktionale Programmierung  
Thema: Funktionale Perlen, Arrays  
ausgegeben: Di, 02.05.2017, fällig: Di, 16.05.2017 (14 Tage!)**

Für dieses Aufgabenblatt sollen Sie Haskell-Rechenvorschriften zur Lösung der im folgenden angegebenen Aufgabenstellungen entwickeln und für die Abgabe in einer Datei namens `AufgabeFFP5.hs` in Ihrem Gruppenverzeichnis ablegen, wie gewohnt auf oberstem Niveau. Kommentieren Sie Ihre Programme aussagekräftig und benutzen Sie, wo sinnvoll, Hilfsfunktionen und Konstanten.

- In Kapitel 4.4 der Vorlesung ist mittels gleichungsbasierten Schließens (“equational reasoning”) ein Algorithmus zur Lösung des MNSS-Problems entwickelt worden, dessen Berechnungsaufwand linear in der Länge der Argumentliste ist. Versuchen Sie auf ähnliche Weise von einer konzeptuell einfachen, möglicherweise ineffizienten, aber “offensichtlich” richtigen Lösungsvariante `simpleMSS :: [Int] -> [Int]` zu einer effiziente(re)n Variante Ihres Verfahrens zur Berechnung der maximalen Abschnittssumme einer Liste zu gelangen, d.h. zu einer effiziente(re)n Lösung `smartMSS :: [Int] -> [Int]` für das MSS-Problem.

Geben Sie also Implementierungen der Haskell-Rechenvorschriften

- `simpleMSS :: [Int] -> [Int]`
- `smartMSS :: [Int] -> [Int]`

an. Gibt es mehr als eine Abschnittssumme, deren Summe maximal ist, so ist es egal, welchen Abschnitt mit maximalem Summenwert Ihre Funktionen zurückliefern.

Testen Sie Ihre beiden *MSS*-Löser anhand selbstgewählter Eingabelisten und vergleichen Sie (ohne Abgabe!) die Performanz Ihrer beiden Löser.

- In dieser Aufgabe beschäftigen wir uns mit einer Knobelaufgabe namens *Zeltlager*. Die linke Abbildung zeigt ein ungelöstes *Zeltlager*-Rätsel, die rechte Abbildung die zugehörige Lösung. Dabei bezeichnet jedes B den Standort eines Baums, jedes Z den Standort eines Zelts im Zeltlager.

a)

	4	0	3	1	3	1	3	1
3		B	B				B	B
1			B					
3								
1	B				B			B
1				B				
3	B			B				B
0	B			B		B		
4		B						

b)

	4	0	3	1	3	1	3	1
3	Z	B	B	Z		Z	B	B
1			B					Z
3	Z		Z		Z			
1	B				B		Z	B
1	Z				B			
3	B		Z	B	Z		Z	B
0	B				B		B	
4	Z	B	Z		Z		Z	

Die Spielregeln von *Zeltlager* sind wie folgt:

- Ein Zeltlager ist durch eine quadratische  $8 \times 8$  Matrix gegeben.
- Neben jedem Baum (B) soll waagrecht oder senkrecht (mindestens) ein Zelt (Z) stehen.
- Die Zahlen oberhalb und links neben der Matrix geben die Gesamtzahl der Zelte in der jeweiligen Zeltlagerspalte bzw. Zeltlagerreihe an.
- Kein Zelt darf ein anderes Zelt berühren, auch nicht diagonal.

Zur Modellierung eines *Zeltlager*-Rätsels benutzen wir folgende Haskell-Typen:

```
data Content = Tree | Tent | Empty deriving (Eq,Ord,Show)
type Camp    = Array (Int,Int) Content
type Row     = Int -- ausschliesslich Werte von 1 bis 8
type Column  = Int -- ausschliesslich Werte von 1 bis 8
type LocationsOfTrees = [(Row,Column)]
type TentsPerRow      = [Int] -- Liste der Laenge 8, ausschliesslich
                               Werte von 0 bis 4; Wert des i-ten
                               Elements bezeichnet Zahl der Zelte
                               in Reihe i
type TentsPerColumn   = [Int] -- Liste der Laenge 8, ausschliesslich
                               Werte von 0 bis 4; Wert des j-ten
                               Elements bezeichnet Zahl der Zelte
                               in Spalte j
```

Schreiben Sie nach dem Vorbild der funktionalen Perlen aus Kapitel 4 der Vorlesung, besonders des *Sudoku*-Lösers aus Kapitel 4.6, zwei Haskell-Rechenvorschriften

- `simpleCamp :: LocationsOfTrees -> TentsPerRow -> TentsPerColumn -> Camp`
- `smartCamp :: LocationsOfTrees -> TentsPerRow -> TentsPerColumn -> Camp`

zur Lösung von *Zeltlager*-Rätseln der Dimension  $8 \times 8$ .

Schreiben Sie zusätzlich eine Ausgabefunktion

- `outCamp :: Camp -> [[Char]]`

die ein Zeltlager in vereinfachter Form ausgibt, d.h. `outCamp` soll Werte des Typs `Camp` in Form von Listen von Listen von Zeichen vom Typ `Char` ausgeben, wobei ausschließlich die Zeichen 'B', 'Z' und 'u' für *Baum*, *Zelt* bzw. *unbesetzt* benutzt werden und die Ergebnisliste in aufsteigender Folge von Reihen die Belegung der verschiedenen Zeltlagerreihen angibt.

Testen Sie Ihre beiden *Zeltlager*-Löser anhand selbstgewählter *Zeltlager*-Rätsel und vergleichen Sie (ohne Abgabe!) die Performanz Ihrer beiden Löser. Geeignete Testrätsel finden Sie zahlreich im Web.

*Hinweise:*

- Sie dürfen davon ausgehen, dass die Funktionen `simpleCamp` und `smartCamp` ausschließlich auf Beschreibungen ‘gültiger’ *Zeltlager*-Rätsel der Dimension  $8 \times 8$  angewendet werden.
- Ist die Lösung eines *Zeltlager*-Rätsels nicht eindeutig, ist es egal, welche Lösung Ihre Verfahren zurückliefern.
- Hat ein *Zeltlager*-Rätsel keine Lösung, so soll das Eingaberätsel umgewandelt in den Typ `Camp` (also ohne die Angabe von Zeltzahlen pro Reihe und Spalte) zurückgegeben werden.