

**7. Übungsaufgabe zu  
Fortgeschrittene funktionale Programmierung  
Thema: Funktionale Perlen  
ausgegeben: Di, 30.04.2013, fällig: Di, 14.05.2013**

Für dieses Aufgabenblatt sollen Sie Haskell-Rechenvorschriften zur Lösung der im folgenden angegebenen Aufgabenstellungen entwickeln und für die Abgabe in einer Datei namens `AufgabeFFP7.hs` in Ihrem Gruppenverzeichnis ablegen, wie gewohnt auf oberstem Niveau. Kommentieren Sie Ihre Programme aussagekräftig und benutzen Sie, wo sinnvoll, Hilfsfunktionen und Konstanten.

In dieser Aufgabe beschäftigen wir uns mit einem Sudoku-ähnlichen Spiel namens Str8ts. Abbildung a) zeigt ein korrekt gelöstes Str8ts-Rätsel, Abbildung b) ein ungelöstes Str8ts-Rätsel.

a)

		8	9		1	2		
5	8	7	6	9	2	3	4	1
7	9	6	8	5		4	3	2
6	7	9	5	4		1	2	3
	4	2	7	3	6	5	1	8
4	3	5	2	8	7			6
2	1	3		6	5	7	8	4
3	2	1	4	7	8	6	9	5
1		4	3		9	8	6	

b)

5			4					7
		9	3					
2					6			
								1
7							3	
4				9				
	6				1			

Die Spielregeln von Str8ts sind wie folgt:

- (i) Die Ziffern 1 bis 9 dürfen pro Zeile und Spalte nur einmal vorkommen.
- (ii) Zusammenhängende weiße Felder enthalten direkt aufeinander folgende Zahlen, die aber in beliebiger Reihenfolge stehen dürfen, sog. *Straßen* (Beispiel: Die Ziffernfolge 5-2-6-4-3 ist eine Straße, die Ziffernfolge 1-2-4 ist keine Straße).
- (iii) Schwarze Felder trennen benachbarte Straßen und sind nicht selber Teil einer Straße.
- (iv) Ziffern in schwarzen Feldern gehören zu keiner Straße, blockieren jedoch diese Ziffern sowohl in der Zeile als auch in der Spalte.

Zur Modellierung eines Str8ts-Rätsels benutzen wir folgende Haskell-Typen:

```
data Color      = Black | White deriving Show
data Digit     = One | Two | Three | Four | Five | Six
               | Seven | Eight | Nine | Blank deriving Show
type Str8ts    = Array (Int,Int) (Color,Digit) deriving Show

type ColorOut  = Char -- Nur 'b' fuer schwarz und 'w' fuer weiss
               -- werden benutzt.
type DigitOut  = Int  -- Nur 1,2,3,4,5,6,7,8,9,0 werden benutzt;
               -- 0 wird dabei als Platzhalter fuer 'blank' benutzt.
type Str8tsOut = Array (Int,Int) (ColorOut,DigitOut)
```

Schreiben Sie nach dem Vorbild der Sudoku-Löser aus Kapitel 4.5 der Vorlesung zwei Haskell-Funktionen

1. `naiveStr8ts :: Str8ts -> Str8tsOut`
2. `fastStr8ts :: Str8ts -> Str8tsOut`

zur Lösung von Str8ts-Rätseln.

Die Funktion `naiveStr8ts` soll dabei nach dem Vorbild des initialen Sudoku-Lösers gemäß Ansatz 2 geschrieben werden (siehe Folien 4 und 5 in Kapitel 4.5). Diese Lösung soll anschließend nach dem Vorbild des Sudoku-Lösers aus Kapitel 4.5 zu einer effizienteren Lösung `fastStr8ts` weiterentwickelt werden.

Testen Sie Ihre beiden Str8ts-Löser anhand selbstgewählter Str8ts-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 `naiveStr8ts` und `fastStr8ts` ausschließlich auf 'gültige' Str8ts-Rätsel der Dimension  $9 \times 9$  angewendet werden, d.h. in keiner Zeile oder Spalte kommt eine Ziffer mehrfach vor.
- Ist die Lösung eines Str8ts-Rätsels nicht eindeutig bestimmt, ist es egal, welche Lösung Ihr Verfahren zurückliefert.
- Hat ein Str8ts-Rätsel keine Lösung, so soll das Eingaberätsel umgewandelt in den Typ `Str8tsOut`, aber ansonsten unverändert zurückgegeben werden.