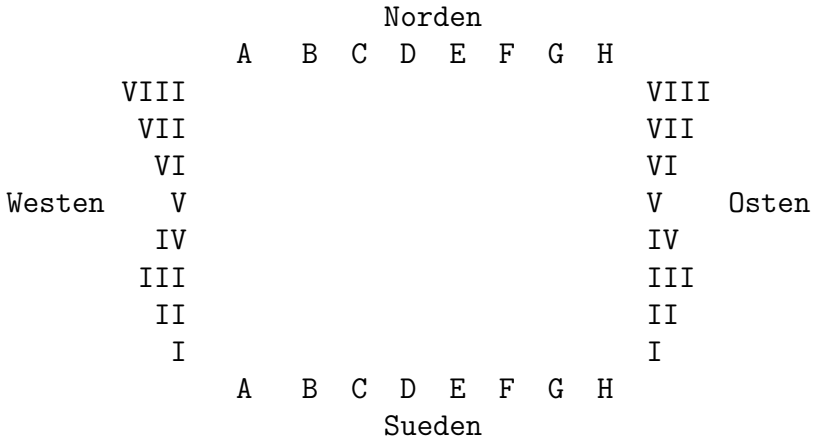
4. Übungsaufgabe zu
Fortgeschrittene funktionale Programmierung
Thema: Funktionale Felder und Monaden
Ausgegeben: Do, 26.04.2018, abzugeben: Mi, 09.05.2018 (15:00 Uhr)

Für dieses Aufgabenblatt sollen Sie Haskell-Rechenvorschriften zur Lösung der im folgenden angegebenen Aufgabenstellungen entwickeln und für die Abgabe in einer Datei namens `AufgabeFFP4.hs` in Ihrem Gruppenverzeichnis ablegen, wie gewohnt auf oberstem Niveau. Kommentieren Sie Ihre Programme aussagekräftig und benutzen Sie, wo sinnvoll, Hilfsfunktionen und Konstanten.

- Türme im Schach dürfen sich nach rechts und links und nach oben und unten bewegen; Läufer diagonal nach oben und unten. Wir können die Turmbewegungen mit den Himmelsrichtungen Nord, Ost, Süd und West beschreiben, die von Läufern mit den Himmelsrichtungen Nordwest, Nordost, Südost und Südwest.



  In dieser Aufgabe soll (in zwei Varianten) eine Funktion geschrieben werden, die für einen Läufer oder Turm auf einem bestimmten Ausgangsfeld dasjenige Feld berechnet, auf dem die Figur nach der Ausführung einer bestimmten Zugfolge zu stehen kommt.

  Eine Zugfolge ist dabei durch eine Liste von Paaren bestehend aus einer Himmelsrichtung (der Zugrichtung) und der Zuglänge (Anzahl Felder) gegeben. Ist die Länge größer oder gleich 0, soll sich die Figur entsprechend viele Felder in der angegebenen Himmelsrichtung bewegen; ist die Länge kleiner als 0 soll sich die Figur in die entgegengesetzte Richtung bewegen, also nach Norden statt Süden oder nach Südosten statt Nordwesten zum Beispiel. Kein Zug einer Figur darf dabei über ein besetztes Feld hinweggehen oder darauf enden. Der Zug endet in solchen Fällen auf dem letztzulässigen Feld davor.

  Zusätzlich zu diesen generellen Zugregeln müssen Turm- und Läuferbewegungen folgende Regeln einhalten:

– *Türme:* Züge enden nicht nur vorzeitig vor einem besetzten Feld, sondern auch wenn das letzte Feld einer Schachbrettzeile oder -reihe erreicht wird.
– *Läufer:* Züge enden zwar vorzeitig vor einem besetzten Feld, nicht aber wenn das Ende einer Zeile oder Reihe erreicht wird. Für einen Läufer stellt sich das Schachbrett als Torus dar. Verlässt ein Läufer das Schachbrett in einer Richtung, so betritt er es auf dem 'gegenüberliegenden' Feld wieder und setzt dort seinen Zug fort. Verlässt etwa ein Läufer auf Feld VIII/B das Schachfeld in nordöstlicher Richtung hin auf das nichtexistente Feld IX/C, so taucht er stattdessen auf dem Feld I/C des Schachbretts wieder auf.
– *Türme* und *Läufer:* Ist das Ausgangsfeld besetzt, wird der Wert `Nothing` des Typs `Maybe Endfeld` zurückgeliefert. Ist eine Zugrichtung einer Figur nicht möglich, d.h. diagonales Ziehen für Türme bzw. gerades Ziehen für Läufer, so wird ein solcher Zug einer Zugfolge nicht beachtet. Der Zug wird übergangen, ohne dass sich die Figur bewegt.

Zur Modellierung der Aufgabe werden folgende Typen verwendet:

```
import Array

-- Modellierung Schachfiguren
data Schachfigur = Turm | Laeufer deriving (Eq,Show)

-- Modellierung Zugrichtung
data Zugrichtung = N | O | S | W
                 | NW | NO | SO | SW deriving (Eq,Show)

-- Modellierung Laenge eines Zugs in Anzahl von Feldern
type Zuglaenge = Int

 -- Modellierung Schachbrett
data Zeile = I | II | III | IV | V | VI | VII | VIII
                                        deriving (Eq,Ord,Enum,Show)
data Reihe = A | B | C | D | E | F | G | H
                                        deriving (Eq,Ord,Enum,Show)
type Besetzt = Bool

instance Ix Zeile where...
instance Ix Reihe where...
type Schachbrett = Array (Reihe,Zeile) Bool

type Schachbrettfeld = (Reihe,Zeile)
type Ausgangsfeld    = Schachbrettfeld
type Endfeld         = Schachbrettfeld
type Zug             = (Zugrichtung,Zuglaenge)
type Zugfolge        = [Zug]
```

```
-- Gesucht sind folgende zwei Funktionen:
fuehre_zugfolge_aus :: Schachfigur -> Schachbrett -> Ausgangsfeld
                                    -> Zugfolge -> Maybe Endfeld
fuehre_zugfolge_aus_mf :: Schachfigur -> Schachbrett -> Ausgangsfeld
                                    -> Zugfolge -> Maybe Endfeld
```

Die Funktion `fuehre_zugfolge_aus` soll die Aufgabe 'herkömmlich' (ohne Monaden) lösen, die Funktion `fuehre_zugfolge_aus_mf` durch Aufruf passender Hilfsfunktionen mithilfe der Zustandsmonade und ggf. Funktoren, so dass die regelgerechte Ausführung von Zügen durch den monadischen Kompositionsoperator `>>=` (und ggf. `fmap`) geleistet wird. Überlegen Sie sich, welche Rolle die Typen Schachfigur, Schachbrett, Ausgangsfeld, Zug und Zugfolge für Funktor und Zustandsmonade spielen. Nehmen Sie, wo nötig `newtype`- oder `data`-Deklarationen vor, um Instanzbildungen vornehmen oder vordefinierte Instanzen für `Functor` und `Monad` ausnutzen zu können. Ergänzen Sie schließlich ggf. nötige Extraktions- und Transformationsfunktionen, um Werte des Ergebnistyps `Maybe Endfeld` zu erhalten. Sie können auch gänzlich neue Typen einführen, wenn nötig oder hilfreich, etwa um Positionsinformation angereicherte Schachfiguren als Paare aus Figur und Position. Lassen Sie sich ggf. vom Roboterbeispiel in Kapitel 16.1 inspirieren.