5. Übungsaufgabe zu
Fortgeschrittene Funktionale Programmierung
Thema: Monadische Programmierung – Robotersimulation
ausgegeben: 03.06.2005, fällig: 09.06.2005

ACHTUNG: Abgabetermine verlängert bis zum 21.06.2005/28.06.2005!

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 Homeverzeichnis ablegen, wie gewohnt auf oberstem Niveau. Kommentieren Sie Ihre Programme wieder aussagekräftig und benutzen Sie, wo sinnvoll, Hilfsfunktionen und Konstanten.

Implementieren Sie eine in Haskell eingebettete imperative Sprache zur Simulation eines einfachen Roboters auf einem ebenen Zeichenbrett. Der Roboter kann sich auf dem unendlich großen Zeichenbrett zweidimensional in vier Richtungen bewegen. Er hat einen Zeichenstift, den er auf das Zeichenbrett aufsetzen oder vom Zeichenbrett abheben kann. Durch Bewegungen mit aufgesetztem Zeichenstift zieht er Linien. Andere Bewegungen hinterlassen keine Spuren. Der Zustand des Roboters besteht aus der aktuellen Position am Zeichenbrett (X- und Y-Koordinate), der Blickrichtung (0, 90, 180 oder 270 Grad ausgehend von der Ursprungsposition entlang der X-Koordinate nach rechts) und dem Zustand des Zeichenstifts (gehoben oder gesenkt). Entwickeln Sie eine Monade `Roboter`, die es Ihnen ermöglicht, Roboter in einem `do`-Ausdruck mit folgenden Befehlen anzusprechen:

```
up   :: Roboter Bool -- hebt Stift ab, True = Stift vorher gesenkt
down :: Roboter Bool -- setzt Stift auf, True = Stift vorher gesenkt
move :: Integer -> Roboter (Integer, Integer)
  -- bewegt um gegebene Schritte (> 0) vorwärts, liefert alte Position
moveTo :: (Integer, Integer) -> Roboter (Integer, Integer)
  -- bewegt zu gegebener Position, gibt alte Position zurück
turn :: Int -> Roboter Int -- dreht Roboter nach rechts
  -- Argument: 0, 90, 180 oder 270 Grad; liefert alte Blickrichtung
turnTo :: Int -> Roboter Int -- dreht in gegebene Blickrichtung
  -- Argument: 0, 90, 180 oder 270 Grad; liefert alte Blickrichtung
```

Jede dieser Funktionen liefert als Ergebnis (gekapselt in `Roboter`) den Zustand der betroffenen Komponente vor Befehlsausführung. Die Funktion `simulate :: Roboter a -> (Integer,Integer,Int,Bool)` soll den Zustand eines Roboters im Ergebnistupel `(x,y,d,p)` zurück geben: `x` und `y` sind die Koordinaten, `d` die Blickrichtung, und `p` besagt, ob der Zeichenstift abgesenkt ist. Im Anfangszustand hat das Tupel den Wert `(0,0,0,False)`. Beispielsweise soll

```
simulate (do {a<-turn 90; move 2; turnTo a; move 3})
```

als Ergebnis `(3,-2,0,False)` liefern.