Eine Sache lernt man, indem man sie macht.
Cesare Pavese (1908-1950)
italien. Schriftsteller

Für das Können gibt es nur einen Beweis, das Tun.
Marie von Ebner-Eschenbach (1830-1916)
österr. Schriftstellerin

7. Aufgabenblatt zu Funktionale Programmierung von Fr, 27.11.2020.
Erstabgabe: Fr, 04.12.2020 (12:00 Uhr)

Themen: *Funktionen als Datenstrukturen (Band, Rechenband), Generatoren und Ströme (Spuren), Rechnen mit Funktionen (Simulator, Spurfunktionen,...) (Kap. 1–14, 18.4)*

- **Teil A, programmiertechnische Aufgaben:** Besprechung am ersten Übungsgruppentermin, der auf die *Zweitabgabe* der programmiertechnischen Aufgaben folgt.
- Teil B, Papier- und Bleistiftaufgaben: Entfällt auf den Angaben 5 bis 7.
- **Teil C, Terminhinweise:** Für Vorlesung, Klein- und Großübungsgruppen.

Wichtig

1. Befolgen Sie die Anweisungen aus den Begleitdateien zu Angabe 1 sorgfältig, um ein reibungsloses Zusammenspiel mit dem Testsystem sicherzustellen. Wenn Sie Fragen dazu haben, stellen Sie diese bitte im TISS-Forum zur Lehrveranstaltung.

2. Erweitern Sie für die für diese Angabe zu schreibenden Rechenvorschriften die zur Verfügung gestellte Rahmendatei

   `Angabe7.hs`

   und legen Sie diese auf oberstem Niveau in Ihrem *home*-Verzeichnis ab. Achten Sie darauf, dass "Gruppe" Leserechte für diese Datei hat. Wenn nicht, setzen Sie diese Leserechte mittels `chmod g+r Angabe7.hs`.

   Löschen Sie keinesfalls eine Deklaration aus dieser *Template*-Datei! Auch dann nicht, wenn Sie einige dieser Deklarationen nicht oder nicht vollständig implementieren wollen. Löschen Sie auch nicht die Modul-Anweisung `module Angabe7 where` am Anfang der *Template*-Datei.

3. Der Name der Abgabedatei ist für Erst- und Zweitabgabe ident!

4. Benutzen Sie keine selbstdefinierten Module! Wenn Sie (für spätere Angaben) einzelne Rechenvorschriften früherer Lösungen wiederverwenden möchten, kopieren Sie diese bitte in die neue Abgabedatei ein. Eine `import`-Anweisung für selbstdefinierte Module schlägt für die Auswertung durch das Abgabesystem fehl, weil Ihre Modul-Datei, aus der importiert werden soll, vom Testsystem nicht mit abgesammelt wird.

5. Ihre Programmierlösungen werden stets auf der Maschine `g0` mit der dort installierten Version von GHCi überprüft. Stellen Sie deshalb sicher, dass sich Ihre Programme (auch) auf der `g0` unter GHCi so verhalten, wie von Ihnen gewünscht, und überzeugen Sie sich bei jeder Abgabe davon! Das gilt besonders, wenn Sie für die Entwicklung Ihrer Haskell-Programme mit einer anderen Maschine, einer anderen GHCi-Version oder/und einem anderen Werkzeug wie etwa Hugs arbeiten!

A Programmiertechnische Aufgaben (beurteilt, max. 100 Punkte)

Erweitern Sie zur Lösung der programmiertechnischen Aufgaben die Rahmendatei `Angabe7.hs`. Kommentieren Sie die Rechenvorschriften in Ihrem Programm zweckmäßig, aussagekräftig und problemangemessen. Benutzen Sie, wo sinnvoll, Hilfsfunktionen und Konstanten. Versehen Sie alle Funktionen, die Sie zur Lösung der Aufgaben benötigen, mit ihren Typdeklarationen, d.h. geben Sie stets deren syntaktische Signatur (kurz: Signatur), explizit an.

In Anhang B.1 der Vorlesungsunterlagen sind Turingmaschinen mit einem unendlichen Rechenband eingeführt.

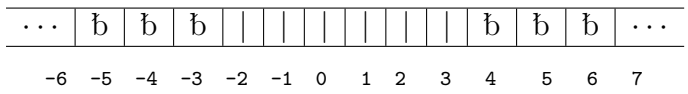
Das unendliche Rechenband einer Turingmaschine kann in einfacher Weise durch eine Funktion ρ auf den ganzen Zahlen in den Zeichenvorrat der Turingmaschine modelliert werden. Dabei liefert ρ für jede Zahl das Zeichen, das auf dem entsprechenden Feld des unendlichen Rechenbands steht. So modelliert die Funktion

$$\rho : \mathbb{Z} \to \{|, \text{ƀ}\}$$

mit

$$\forall\, z \in \mathbb{Z}.\ \rho(z) =_{df} \begin{cases} | & \text{falls } -3 \leq z \leq 2 \\ \text{ƀ} & \text{sonst} \end{cases}$$

das unendliche Rechenband, auf dem auf den Feldern -3, -2, -1, 0, 1, 2 das Zeichen | steht, auf allen anderen Feldern das Leerzeichen, dargestellt durch das Symbol ƀ (für 'blank'):



Unendliches Rechenband

In Haskell können wir den Zahlbereich $\mathbb{Z}$ durch den (jedenfalls konzeptuell) nicht beschränkten Datentyp `Integer` implementieren, den Zeichenvorrat einer Turingmaschine durch Haskells Zeichentyp `Char`, das Leerzeichen ƀ durch einen speziellen Konstruktorwert `Blank` und Rechenbänder (i.w.) durch Funktionen auf `Integer` in den Zeichenvorrat `Char` und den Spezialwert `Blank`. Bandfelder mit einem Wert aus `Char` heißen *beschrieben*, Felder mit dem Wert `Blank` heißen *unbeschrieben*.

```
-- Natuerliche Zahlen, ganze Zahlen
type Nat0 = Integer
type Zett = Integer

-- Zeichenvorrat, Bandalphabet einer Turingmaschine
type Zeichenvorrat = Char
data Bandalphabet  = Z Zeichenvorrat
                   | Blank deriving (Eq,Show)

-- Rechenband
type Min      = Zett -- Kleinster Index eines beschriebenen Bandfelds
type Max      = Zett -- Groesster Index eines beschriebenen Bandfelds
data MinMax   = B Min Max                  -- B wie `Beschrieben'
              | U deriving (Eq,Show)       -- U wie `Unbeschrieben'
type Bandfeld = Zett
```

```
type Band       = (Bandfeld -> Bandalphabet)
data Rechenband = RB MinMax Band

-- Lese- und Schreibkopf (LSK)
type LSK_Position      = Zett
type Zeichen_unter_LSK = Bandalphabet
data Richtung          = Links | Rechts deriving (Eq,Show)
data Befehl            = Drucke Bandalphabet
                       | Bewege_LSK_nach Richtung deriving (Eq,Show)

 -- Interne Turingmaschinenzustaende
type Zustand               = Nat0
type Interner_Zustand      = Zustand
type Interner_Folgezustand = Zustand

-- Abkuerzungen
type LSKZ = Zeichen_unter_LSK
type IZ   = Interner_Zustand
type IFZ  = Interner_Folgezustand

-- Turing-Tafeln
type Zeile       = (IZ,LSKZ,Befehl,IFZ)
type Turingtafel = [Zeile]

-- Globale Turingmaschinenzustaende
data GZustand = GZ Turingtafel Rechenband IZ LSK_Position

-- Turingmaschinensimulatoreingabe
type Initiales_Rechenband = Rechenband
data Sim_Eingabe = SE Turingtafel Initiales_Rechenband

-- Turingmaschinensimulatorausgabe
type Finaler_interner_Zustand = Zustand
type Finale_LSK_Position      = LSK_Position
type Finales_Rechenband       = Rechenband

-- Abkuerzungen
type FIZ   = Finaler_interner_Zustand
type FLSKP = Finale_LSK_Position
type FRB   = Finales_Rechenband

-- Simulatorausgabe
data Sim_Ausgabe = SA FIZ FLSKP FRB

-- Zulaessige Turingtafeln
ist_zulaessige_Turingtafel :: Turingtafel -> Bool
ist_zulaessige_Turingtafel ...

-- Turingmaschinenzustandsuebergangsfunktion, kurz Transitionsfunktion
transition :: GZustand -> GZustand
transition ...
```

```
-- Spurfunktionen
type Spur = [GZustand]
spur :: GZustand -> Spur
spur ...
zeige_zustand :: GZustand -> String
zeige_zustand ...
zeige_spur :: Spur -> String
zeige_spur ...

-- Turingmaschinensimulator, kurz Simulator
sim :: Sim_Eingabe -> Sim_Ausgabe
sim ...

-- Komfortfunktion zur Vereinfachung der Turingmaschineneingabe
wandle_in_rb :: [Zeichenvorrat] -> Rechenband
```

A.1 Schreiben Sie zwei Haskell-Rechenvorschriften zur Änderung bzw. Aktualisierung von Bändern und Rechenbändern:

```
akt_band :: Band -> Bandfeld -> Bandalphabet -> Band
akt_rechenband :: Rechenband -> Bandfeld -> Bandalphabet -> Rechenband
```

Angewendet auf ein Band bzw. Rechenband, ein Bandfeld und einen Bandalphabetwert liefern `akt_band` bzw. `akt_rechenband` einen Band- bzw. Rechenbandwert, in dem das bezeichnete Bandfeld den neuen Bandalphabetwert trägt, alle anderen Bandfelder denselben Wert wie das entsprechende Feld des Argumentbands bzw. rechenbands. Die Funktion `akt_rechenband` passt darüberhinaus auch den `MinMax`-Wert des Rechenbandwerts entsprechend der Änderung an. Im Spezialfall, dass das Bandfeld mit dem Leerzeichen überschrieben wird und sein Index mit dem `Max`- oder/und `Min`-Wert übereinstimmt, werden diese Werte so angepasst, dass `Max`- und `Min`-Wert wieder den größten bzw. kleinsten Index eines beschriebenen Bandfelds bezeichnen; ist das letzte beschriebene Feld mit dem Leerzeichen überschrieben worden, ist `U` der neue `MinMax`-Wert. Für die Implementierung darf vorausgesetzt werden, dass `akt_rechenband` nur mit stimmigen Argumenten aufgerufen wird, d.h.: (a) der Bandwert des Rechenbandarguments ist das vollständig unbeschriebene Band gdw. der `MinMax`-Wert `U` ist. (b) Ist der `MinMax`-Wert von `U` verschieden, ist (b1) der `Min`-Wert min kleiner oder gleich dem `Max`-Wert max, (b2) die Bandfelder mit den Indizes min und max sind beschrieben und (b3) es gibt ein Bandfeld mit Index m mit $min \leq m \leq max$, das ebenfalls beschrieben ist (beachte: es kann gelten $min = m = max$!).

A.2 Vervollständigen Sie die Implementierungen der Rechenvorschriften und Instanzdeklaration. Nehmen Sie bei Bedarf weitere Zuordnungen von Typen zu Typklassen vor; mit `deriving`-Klauseln, wo möglich, mit `instance`-Deklarationen, wo nicht.

(a) `wandle_in_rb :: [Zeichenvorrat] -> Rechenband`

(b) `ist_zulaessige_Turingtafel :: Turingtafel -> Bool`

(c) `transition :: GZustand -> GZustand`

(d) `spur :: GZustand -> Spur`

(e) `zeige_zustand :: GZustand -> String`

(f) `zeige_spur :: Spur -> String`

(g) `sim :: Sim_Eingabe -> Sim_Ausgabe`

(h) `instance Show Sim_Ausgabe where`

Dabei soll gelten:

(a) *Komforteingabe:* Die Funktion `wandle_in_rb` dient zum einfacheren Aufruf des Turingmaschinensimulators; sie wandelt dazu ihr Argument in offensichtlicher Weise in den entsprechenden Rechenbandwert um. Im Detail: Angewendet auf ein nichtleeres Wort $c_1c_2 \ldots c_k$, $k \geq 1$, über dem Zeichenvorrat der Turingmaschine liefert `wandle_in_rb` den Rechenbandwert mit `Min`-Wert 1, `Max`-Wert k und Bandwert ρ mit $\rho(z) = c_z$ für $1 \leq z \leq k$ und $\rho(z) =$ `Blank` sonst; angewendet auf das leere Wort liefert `wandle_in_rb` den Rechenbandwert mit `MinMax`-Wert `U` und Bandwert ρ mit $\rho(z) =$ `Blank`, $z \in \mathbb{Z}$.

(b) *Zulässige Turingtafel* (`ist_zulaessige_Turingtafel`)*:* Eine Turingtafel ist *zulässig* gdw.: (i) Die Turingtafel ist nicht leer. (ii) Sind `(iz,z,b,ifz)`, `(iz',z',b',ifz')` zwei verschiedene Zeilen der Turingtafel, so gilt: `iz` $\neq$ `iz'` oder `z` $\neq$ `z'`.

(c) *Zustandsübergangsfunktion* (`transition`)*:* Angewendet auf einen Turingmaschinenzustand liefert `transition` den Nachfolgezustand entsprechend der Beschreibung der Arbeitsweise einer Turingmaschine in Anhang B.1. Führt die Maschine dabei den Befehl "Drucke <Zeichen>" aus, wird neben dem Bandwert des Rechenbands auch sein `MinMax`-Wert angepasst, wenn nötig. Ist kein Zustandsübergang möglich (weil die Turingtafel keine Zeile für das Paar aus aktuellem internen Zustand und Zeichen unter dem Lese/Schreibkopf enthält), so bildet die Transitionsfunktion den Argumentzustand ident ab.

(d) *Spur* (`spur`)*:* Angewendet auf einen Anfangszustand der Turingmaschine liefert `spur` die (möglicherweise nicht endliche) Liste von Folgezuständen der Turingmaschine, die die Transitionsfunktion angesetzt auf den Anfangszustand erzeugt; der Anfangszustand ist dabei der erste Zustand der Spur. Die Funktion `spur` terminiert, wenn der aktuelle interne Zustand und das aktuelle Zeichen unter dem Lese/Schreibkopf nicht als Anfang einer Zeile in der Turingtafel vorkommen, wodurch sich die Turingmaschine selbstständig abschaltet (siehe Arbeitsweise einer Turingmaschine in Anhang B.1). Dieser Zustand ist der letzte Zustand der Spur.

(e) *Zustandsausgabe* (`zeige_zustand`)*:* Die Funktion `zeige_zustand` stellt einen Turingmaschinenzustand als Zeichenreihe dar, die den internen Zustand, die Lese/Schreibkopfposition und den Inhalt des beschriebenen Bandbereichs (von *min* bis *max*) beschreibt sowie den kleinsten und größten Index des beschriebenen Bandbereichs; ist das gesamte Band unbeschrieben, wird dies durch `"unbeschrieben"` dargestellt.

Aufrufbeispiele:

```
tt :: Turingtafel
tt = [(0,Blank,Drucke (Z `|`),1),(1,Z `|`,Bewege_LSK_nach Links,2)]
rb1 :: Rechenband
rb1 = wandle_in_rb "|||"
z1 :: GZustand
z1 = GZ tt rb1 1 0
zeige_zustand z1 ->> "(IZ:1,LSK:0,B:|||,Min:1,Max:3)"
```

```
rb2 :: Rechenband
rb2 = wandle_in_rb ""
z2 :: GZustand
z2 = GZ tt rb2 0 1
zeige_zustand z2 ->> "(IZ:0,LSK:1,B:unbeschrieben)"
```

(f) *Spurausgabe* (`zeige_spur`)*:* Die Funktion `zeige_spur` zeigt einen Spurwert als Zeichenreihe. Ein Spurwert wird dabei als Folge von Turingmaschinenzuständen dargestellt, jeweils getrennt durch `" ->> "` (d.h. je ein Leerzeichen vor und nach dem Pfeil).

Aufrufbeispiele:

```
spur1 :: Spur
spur1 = [z1,z1,z1]
text1 = "(IZ:1,LSK:0,B:|||,Min:1,Max:3)"
zeige_spur spur1 ->> text1 ++ " ->> " ++ text1 ++ " ->> " text1
spur2 :: Spur
spur2 = [z1,z2]
text2 = "(IZ:0,LSK:1,B:unbeschrieben)"
zweige_spur spur2 ->> text1 ++ " ->> " ++ text2
```

(g) *Turingmaschinensimulator* (`sim`)*:* Angewendet auf eine Eingabe und eine zulässige Turingtafel simuliert `sim` die Arbeitsweise der Turingmaschine wie in Anhang B.1 beschrieben für diese Eingabe. Terminiert die (Simulation der) Turingmaschine, d.h. ist die Spur, die die Turingmaschine angesetzt auf Turingtafel- und Rechenbandargument der Eingabe erzeugt, endlich, so liefert `sim` den letzten Zustand der erzeugten Spur als Wert vom Typ `Sim_Ausgabe`; ist die erzeugte Spur nicht endlich, so terminiert `sim` nicht.

Aufrufbeispiele:

```
minmax :: MinMax
minmax = B 0 2
rho :: Band
rho = \z -> if (0 <= z) && (z <= 2) then (Z `|`) else Blank
rho_fin :: Rechenband
rb_fin = RB minmax rho
se :: Sim_Eingabe
se = SE [(0,Blank,Drucke (Z `|`),1),[(1,Z `|`,Bewege_LSK_nach Links,2)]
        (wandle_in_rb "||")
sim se ->> (SA 2 -1 rb_fin)
```

(h) *Simulatorausgabe* (`instance Show Sim_Ausgabe`)*:* Machen Sie `Sim_Ausgabe` zu einer Instanz von `Show`. `Sim_Ausgabe`-Werte sollen dabei wie nachstehend gezeigt als Zeichenreihen dargestellt werden. Nichtleeren Bändern wird dabei der kleinste/größte Index eines beschriebenen Bandfelds voran-/nachgestellt; verschiedene Abschnitte der Ausgabe werden durch je ein Leerzeichen getrennt:

```
sa1 :: Sim_Ausgabe
sa1 = SA 2 4 rb1
show sa1 ->> "IZ: 2 // LSKP: 4 // BI: 1>|||<3"
sa2 :: Sim_Ausgabe
sa2 = SA 4 2 rb2
```

```
show sa2 ->> "IZ: 4 // LSKP: 2 // BI: Leer"

show (sim se) ->> "IZ: 2 // LSKP: -1 // BI: 0>|||<2"
```

A.3 **Ohne Beurteilung:** Geben Sie zu allen Rechenvorschriften einen Kommentar an, in dem knapp, aber gut nachvollziehbar beschrieben ist, wie die jeweilige Rechenvorschrift vorgeht.

A.4 **Ohne Abgabe, ohne Beurteilung:** Testen Sie alle Funktionen auch mit weiteren eigenen Testdaten.

B Papier- und Bleistiftaufgaben

Entfällt auf den Angaben 5 bis 7.

Iucundi acti labores.
Getane Arbeiten sind angenehm.

Cicero (106 - 43 v.Chr.)
röm. Staatsmann und Schriftsteller

C Terminhinweise

1. **Nächste Vorlesungstermine: Mittwoch, 02./16.12.2020**, 08:15-09:45 Uhr, Echtzeitvideokonferenz unter der bekannten Zoom-URL (siehe TISS):
   - 08:15-09:00 Uhr: Vorlesungsteil VI bzw. VII
   - 09:15-09:45 Uhr: Umgekehrtes Klassenzimmer zu Vorlesungsteil V bzw. VI
2. **Nächste Kleinübungsgruppentermine:** Die Kleinübungsgruppentreffen finden wie geplant statt, ausschließlich *online* als Echtzeitvideokonferenzen.
   - **Zoom-URLs für alle KÜGs: Siehe Tuwel!**
   - KÜG 1&7: Di, 01./08./15.12.2020, 16:00-17:00 Uhr, online via Zoom
   - KÜG 2&8: Do, 03./10./17.12.2020, 08:00-9:00 Uhr, online via Zoom
   - KÜG 3&9: Do, 03./10./17.12.2020, 11:00-12:00 Uhr, online via Zoom
   - KÜG 4&10: Do, 03./10./17.12.2020, 14:00-15:00 Uhr, online via Zoom
   - *Fr, 04./11./18.12.2020, 08:00-9:00 Uhr: Termine entfallen!*
   - KÜG 5&6, KÜG 11&12: Fr, 04./11./18.12.2020, 11:00-12:00 Uhr, online via Zoom
   - KÜG 13: Mi, 02./09./16.12.2020, 15:00-16:00 Uhr, online via Zoom
   - KÜG 14: Fr, 04./11./18.12.2020, 15:00-16:00 Uhr, online via Zoom
   - *Fr, 04./11./18.12.2020, 17:00-18:00 Uhr: Termine entfallen!*
3. **Nächste Plenumsübungsgruppentermine:** Beide Plenumsübungsgruppentreffen finden wie geplant statt, ausschließlich *online* als Echtzeitvideokonferenzen.
   - **Zoom-URLs für beide PÜGs: Siehe Tuwel!**
   - PÜG I: Di, 01./08./15.12.2020, 09:00-10:00 Uhr, online via Zoom
   - PÜG II: Mi, 02./09./16.12.2020, 16:00-17:00 Uhr, online via Zoom