9. Aufgabenblatt zu Funktionale Programmierung vom 12.12.2012.
Fällig: 09.01.2013 / 16.01.2013 (jeweils 15:00 Uhr)

Themen: *Funktionen auf Graphen und Strömen*

Für dieses Aufgabenblatt sollen Sie Haskell-Rechenvorschriften für die Lösung der unten angegebenen Aufgabenstellungen entwickeln und für die Abgabe in einer Datei namens `Aufgabe9.hs` ablegen. Sie sollen für die Lösung dieses Aufgabenblatts also ein "gewöhnliches" Haskell-Skript schreiben. Versehen Sie wieder wie auf den bisherigen Aufgabenblättern alle Funktionen, die Sie zur Lösung brauchen, mit ihren Typdeklarationen und kommentieren Sie Ihre Programme aussagekräftig. Benutzen Sie, wo sinnvoll, Hilfsfunktionen und Konstanten.

Im einzelnen sollen Sie die im folgenden beschriebenen Problemstellungen bearbeiten.

1. Weihnachten rückt näher und der Weihnachtsmann hat seine Reiseplanung noch nicht abgeschlossen, sondern noch mehr Fragen. Helfen Sie mit weiteren Haskell-Rechenvorschriften, ihm auch auf die neuen Fragen Antworten zu geben. Wir benötigen dazu wieder die Datentypen von Aufgabenblatt 8:

```
type Country    = String
type Countries  = [Country]
type TravelTime = Integer  -- Travel time in minutes
data Connection = Air Country Country TravelTime
                  | Sea Country Country TravelTime
                  | Rail Country Country TravelTime
                  | Road Country Country TravelTime deriving (Eq,Ord,Show)
type Connections = [Connection]
data Itinerary   = NoRoute | Route (Connections,TravelTime) deriving (Eq,Ord,Show)
```

   - Welche Länder sind von Land A ohne Luft- und Seeetappen aus erreichbar?
     `yieldGroundReachable :: Connections -> Country -> Countries`
   - Welche Länder sind von Land A aus gar nicht erreichbar?
     `yieldUnreachable :: Connections -> Country -> Countries`
   - Liefere alle in Land A beginnenden Routen, auf denen jedes Land genau einmal besucht wird, falls es solche gibt, zusammen mit der jeweiligen Reisezeit; ansonsten die einelementige Liste mit Element `NoRoute`. Enthält die Ergebnisliste mehr als einen Reiseplan, soll sie entsprechend der auf `Itinerary` durch die `deriving`-Klausel festgelegten Ordnung aufsteigend sortiert sein.
     `yieldCompleteTrips :: Connections -> Country -> [Itinerary]`
   - Gibt es eine Route von Land A aus, auf der jedes Land genau einmal besucht wird und die wieder in Land A endet?
     `isRoundTrip :: Connections -> Country -> Bool`

   Dabei gilt: Zwischen je zwei Ländern kann es keine, eine oder auch mehr als eine Verbindung geben. Es kann auch mehr als je eine Flug-, See-, Bahn- oder Busverbindung zwischen zwei Ländern geben. Mit jeder Verbindung von Land A nach Land B ist zugleich auch die Rückverbindung mit gleicher Reisezeit gegeben. Ausgangs- und Zielland in einer Verbindung dürfen übereinstimmen; solche Inlandsverbindungen bringen den Weihnachtsmann auf dem Weg zu einem davon abweichenden Zielland allerdings nicht weiter. Auch sonst wird dem Weihnachtsmann nichts geschenkt. Nicht einmal Zeit. Deshalb ist bei möglicherweise negativ angegebenen Reisezeiten in `Connection`-Werten oder Funktionsaufrufen mit dem Betrag des entsprechenden `TravelTime`-Werts zu rechnen. Weiters gilt, dass alle Länder implizit durch ihr Vorkommen in mindestens einer der Verbindungen vom Wert `Connections` gegeben sind.

2. Silvester steht vor der Tür. Neben Hasenpfoten, Hufeisen und vierblättrigen Kleeblättern sind auch bestimmte Zahlen in besonderer Weise glücksverheißend. Diese Zahlen bilden einen Strom, den Strom der Glückszahlen.

   Der Strom der *Glückszahlen* ist dabei nach dem Vorbild des *Siebs des Eratosthenes* in folgender Weise gebildet:

   (a) Schreibe alle ungeraden Zahlen beginnend mit 1 auf:
   $1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, \ldots$

   (b) Markiere 3 als erste Glückszahl, streiche dann jede 3te-Zahl aus obigem Strom und erhalte:
   $1, 3, 7, 9, 13, 15, 19, 21, 25, 27, 31, 33, 37, 39, 43, \ldots$

   (c) Markiere die kleinste (von 1 verschiedene) noch nicht markierte Zahl n im Strom als Glückszahl (im ersten Durchlauf die 7) und streiche dann jedes n-te Element aus dem Strom (im ersten Durchlauf also jedes 7-te Element und erhalte $1, 3, 7, 9, 13, 15, 21, 25, 27, 31, 33, 37, 43, \ldots$)

   (d) Gehe zurück zu Schritt (c).

   Alle von 1 verschiedenen Zahlen, die auf diese Weise nicht ausgesiebt werden, sind Glückszahlen.

   Geben Sie eine Haskell-Rechenvorschrift `luckyNumbers :: [Integer]` an, die den mit 3 beginnenden Strom der Glückszahlen liefert (1 wird nicht als Glückszahl angesehen, genausowenig wie 1 als Primzahl angesehen wird. Somit gilt: `take 5 luckyNumbers == [3,7,9,13,15]`).

3. Schreiben Sie Haskell-Rechenvorschriften für folgende Aufgaben:

   - Ist n eine Glückszahl?
     `isLuckyNumber :: Integer -> Bool`
   - Liefere alle Glückszahlen zwischen m und n, m und n einschließlich:
     `yieldLuckyNumbers :: Integer -> Integer -> [Integer]`
     Für $m > n$, wird die leere Liste als Resultat zurückgegeben.
   - Ist n ein Glückszahlzwilling (n heißt *Glückszahlzwilling*, wenn n und mindestens auch $n + 2$ oder $n - 2$ Glückszahlen sind)?
     `isTwinLuckyNumber :: Integer -> Bool`

   Bemerkung: Glückszahlen weisen hinsichtlich Verteilung, Vorkommen von Zwillingen, etc. ähnliche Eigenschaften wie Primzahlen auf.

Hinweis:

- Verwenden Sie *keine* selbst erstellten Module. Wenn Sie Funktionen früherer Lösungen wiederverwenden möchten, kopieren Sie diese in die Abgabedatei `Aufgabe9.hs`. Andere Dateien als diese werden vom Abgabeskript ignoriert.

Haskell Live

Am Freitag, den 14.12.2012, ist der nächste und letzte *Haskell Live*-Termin vor Weihnachten. Der *Haskell Live*-Termin vom 21.12.2012 wird auf Jänner verschoben und auch der Klausurvorbereitung dienen. Der Termin und Raum dafür stehen noch nicht fest. Daneben wird es im Jänner die beiden *Haskell Live*-Termine am Freitag, den 11.01.2013, und am Freitag, den 18.01.2013, geben.

Frohe Weihnachten
und ein erfolgreiches Neues Jahr!