2. Aufgabenblatt zu Funktionale Programmierung vom 12.10.2010. Fällig: 19.10.2010 / 26.10.2010 (jeweils 15:00 Uhr)

Themen: *Funktionen über ganzen Zahlen, Wahrheitswerten, Listen und Tupeln*

Für dieses Aufgabenblatt sollen Sie Haskell-Rechenvorschriften für die Lösung der unten angegebenen Aufgabenstellungen entwickeln und für die Abgabe in einer Datei namens `Aufgabe2.hs` ablegen. Sie sollen für die Lösung dieses Aufgabenblatts also ein "gewöhnliches" Haskell-Skript schreiben. Versehen Sie wieder alle Funktionen, die Sie zur Lösung brauchen, mit ihren Typdeklarationen und kommentieren Sie Ihre Programme aussagekräftig. Benutzen Sie, wo sinnvoll, Hilfsfunktionen und Konstanten.

- Seien m, n natürliche Zahlen, $m, n > 0$. Ein einfacher Test, ob m durch n ohne Rest teilbar ist, kann durch wiederholtes Abziehen von n realisiert werden. Dabei wird n anfänglich von m, anschließend vom jeweiligen Differenzwert abgezogen. m ist genau dann durch n ohne Rest teilbar, wenn 0 ein Element der Folge der Differenzwerte ist. Wir nennen diese (nichtendliche) Folge die *Differenzwertefolge von m und n.*
- Eine natürliche Zahl m ist genau dann durch 7 teilbar, wenn die alternierende Summe $as(m)$ der von rechts genommenen 3er-Blöcke von Ziffern durch 7 teilbar ist. Dabei ist der letzte Ziffernblock ggf. um führende Nullen erweitert zu denken. Z.B. ist die Zahl $m = 24.889.375$ durch 7 teilbar, da $as(24.889.375) = 375 - 889 + 024 = -490 = -70 * 7$. Wir nennen die Folge dieser 3er-Blöcke die *Dreierfolge von m*. Die Dreierfolge von 24.889.375 ist (ohne führende Nullen) die Zahlenfolge $375, 889, 24$.

Im einzelnen sollen Sie nun folgende Problemstellungen bearbeiten:

1. Schreiben Sie eine Haskell-Rechenvorschrift `diffFolge` mit der Signatur `diffFolge :: (Integer,Integer) -> [Integer]`, die angewendet auf das Argumentpaar (m, n), $m, n > 0$, die (Anfangs-) Folge der Differenzwerte von m und n liefert. Das erste Element der Resultatliste ist dabei m, das vorletzte Element stets echt größer als 0 und das letzte Element entweder 0 oder ein Wert echt kleiner als 0.
2. Schreiben Sie eine Haskell-Rechenvorschrift `teilt` mit der Signatur `teilt :: (Integer,Integer) -> Bool`, die angewendet auf das Argumentpaar (m, n), $m, n > 0$, angibt, ob m ohne Rest durch n teilbar ist. Stützen Sie die Implementierung der Funktion `teilt` auf die Funktion `diffFolge` ab, d.h. `teilt` ruft `diffFolge` auf.
3. Schreiben Sie eine Haskell-Rechenvorschrift `zahlenBlock` mit der Signatur `zahlenBlock :: Integer -> [Integer]`, die angewendet auf ein Argument m, $m > 0$, die Dreierfolge von m in Form einer Liste ganzer Zahlen ausgibt. Angewendet auf 24.889.375 liefert `zahlenBlock` also die Resultatliste `[375,889,24]`.
4. Schreiben Sie analog zur Rechenvorschrift `zahlenBlock` eine Haskell-Rechenvorschrift `zeichenreihenBlock` mit der Signatur `zeichenreihenBlock :: Integer -> [String]`, die angewendet auf ein Argument m, $m > 0$, die Dreierfolge von m in Form einer Liste von Zeichenreichen jeweils der Länge 3 ausgibt. Angewendet auf 24.889.375 liefert `zeichenreihenBlock` also die Resultatliste `["375","889","024"]`.
5. Schreiben Sie eine Haskell-Rechenvorschrift `siebenTeilbar` mit der Signatur `siebenTeilbar :: Integer -> Bool`, die feststellt, ob das Argument m, $m > 0$, angibt, ob m ohne Rest durch 7 teilbar ist. Schreiben Sie dazu eine Haskell-Rechenvorschrift `as` mit Signatur `as :: [Integer] -> Integer`, die die alternierende Summe der Argumentliste berechnet. Angewendet auf die Liste `[375,889,24]` liefert `as` also den Wert -490. Stützen Sie anschließend die Implementierung von `siebenTeilbar` auf die Funktionen `zahlenBlock`, `as` und `teilt` ab. Beachten Sie dabei, dass `teilt` nur für positive Argumente wohldefiniert ist und überlegen Sie sich deshalb eine Lösung für den Umgang mit möglicherweise negativen Argumenten.

Sie können davon ausgehen, dass alle Funktionen nur mit "passenden" positiven Argumenten bzw. Argumentlisten aufgerufen werden.

Zusatzaufgabe ohne Abgabe und Punkte: Vergleichen Sie die Auswertungsgeschwindigkeit der Rechenvorschrift `siebenTeilbar` mit derjenigen der Rechenvorschrift `teilt` für den Teiler 7. Ist mit wachsender Argumentgröße ein Unterschied zu bemerken? Wenn ja, ab ungefähr welcher Argumentgröße wird der Unterschied augenfällig?

Wichtiger Hinweis: Wenn Sie einzelne Rechenvorschriften aus früheren Lösungen wieder verwenden möchten, so kopieren Sie diese in die neue Abgabedatei ein. Ein `import` schlägt für die Auswertung durch das Abgabeskript fehl, weil Ihre alte Lösung, aus der importiert wird, nicht mit abgesammelt wird. Dehalb: Kopieren statt importieren zur Wiederwendung!

Haskell Live

Am Freitag, den 15.10.2010, werden wir uns in *Haskell Live* u.a. mit der Aufgabe *"Licht oder nicht Licht - Das ist hier die Frage!"* beschäftigen.

Licht oder nicht Licht - Das ist hier die Frage!

Zu den Aufgaben des Nachtwachdienstes an unserer Universität gehört das regelmäßige Ein- und Ausschalten der Korridorbeleuchtungen. In manchen dieser Korridore hat jede der dort befindlichen Lampen einen eigenen Ein- und Ausschalter und jedes Betätigen eines dieser Schalter schaltet die zugehörige Lampe ein bzw. aus, je nachdem, ob die entsprechende Lampe vorher aus- bzw. eingeschalten war. Einer der Nachtwächter hat es sich in diesen Korridoren zur Angewohnheit gemacht, die Lampen auf eine ganz spezielle Art und Weise ein- und auszuschalten: Einen Korridor mit n Lampen durchquert er dabei n-mal vollständig hin und her. Auf dem Hinweg des i-ten Durchgangs betätigt er jeden Schalter, dessen Position ohne Rest durch i teilbar ist. Auf dem Rückweg zum Ausgangspunkt des i-ten und jeden anderen Durchgangs betätigt er hingegen keinen Schalter. Ein *Durchgang* ist also der Hinweg unter entsprechender Betätigung der Lichtschalter und der Rückweg zum Ausgangspunkt ohne Betätigung irgendwelcher Lichtschalter.

Die Frage ist nun folgende: Wenn beim Eintreffen des Nachtwächters in einem solchen Korridor alle n Lampen aus sind, ist nach der vollständigen Absolvierung aller n Durchgänge die n-te und damit letzte Lampe im Korridor an oder aus?

Schreiben Sie ein Programm in Haskell oder in irgendeiner anderen Programmiersprache ihrer Wahl, das diese Frage für eine als Argument vorgegebene positive Zahl von Lampen im Korridor beantwortet.

Für n gleich 3 oder n gleich 8191 sollte Ihr Programm die Antwort "aus" liefern, für n gleich 6241 die Antwort "an".