Technische Universität München
Institut für Informatik
Prof. Tobias Nipkow, Ph.D.
Lars Noschinski, Lars Hupel, Dr. Jasmin Blanchette

WS 2013/14
17.12.2014
Abgabe: 07.01.2014, 15:30

Einführung in die Informatik 2
10. Übung

Aufgabe G10.1 Dem Zufall auf die Sprünge helfen

Gesucht ist eine Funktion `nRandomR :: (Int, Int) -> Int -> IO [Int]`, die eine Liste von Zufallszahlen generiert. Die Ausgabeliste von `nRandomR (`l`,`h`)` n soll n Elemente enthalten, wobei jedes dieser Elemente $\geq l$ und $\leq h$ sein soll. Außerdem soll die Ausgabeliste keine Elemente doppelt enthalten. Falls in dem vorgegebenen Bereich `(`l`, `h`)` nicht genügend unterschiedliche Zahlen sind, muss Ihre Implementierung nicht terminieren (z.B. für `nRandomR (1,2) 3`).

Verwenden Sie die Funktion `randomRIO :: (Int, Int) -> IO Int` aus `System.Random`, um eine einzige Zufallszahl aus dem vorgegebenen Bereich zu erzeugen. Duplikate vermeiden Sie mit einem Hilfsargument für bereits "gewürfelte" Zahlen. So erkennen Sie Wiederholungen und können einfach neu würfeln.

Aufgabe G10.2 Zahlenraten

Wir wollen folgendes Spiel in Haskell implementieren: Zunächst wählt das Programm eine zufällige Zahl zwischen 0 und 100 (einschließlich) und fragt dann den Benutzer nach einer Zahl. Hat der Benutzer die richtige Zahl erraten, gibt das Programm die Anzahl der Versuche zurück. Andernfalls zeigt es an, ob die gewählte Zahl größer oder kleiner ist und fragt erneut nach einer Eingabe, bis der Benutzer die richtige Zahl erraten hat.

1. Schreiben Sie zunächst eine IO-Aktion `getLineInt :: IO Int`, die eine Zeile von der Tastatur einliest. Repräsentiert diese Zeile eine (positive) Zahl, so soll die Zahl zurückgegeben werden, andernfalls soll eine Fehlermeldung ausgegeben werden und erneut eine Eingabe gelesen werden. Dies wiederholt sich solange, bis es eine gültige Eingabe gibt.

   Verwenden Sie die IO-Aktion `getLine :: IO String` um eine Zeile zu lesen. Mit der Funktion `readMaybe :: String -> Maybe Int` aus dem Modul `Text.Read` können Sie eine Eingabe in eine Zahl umwandeln.

2. Verwenden Sie die vorherige Teilaufgabe um das Spiel als IO-Aktion `guessNum :: IO Int` zu implementieren.

Hinweis: Die Aufgaben zum Thema IO sind leicht manuell und schwer automatisch zu testen. Deswegen gibt es auf diesem Übungsblatt nicht für alle Aufgaben QuickCheck-Tests. Testen Sie Ihr Programm vor der Abgabe!

Aufgabe H10.1 Zähle zeilenweise zig Zahlen (3 Punkte)

Schreiben Sie eine IO-Aktion `countNumbers :: IO Int`, die zeilenweise solange Text einliest, bis eine leere Zeile eingegeben wurde. Daraufhin soll ausgegeben werden, wie viele der eingegebenen Zeilen nur eine Zahl enthalten (eine Zahl ist alles, was von `readMaybe :: String -> Maybe Int` als solche erkannt wird; `readMaybe` ist im Modul `Text.Read` definiert). Benutzen Sie als einzige vordefinierte IO-Aktion `getLine :: IO String`.

Beispiel: Wird `countNumbers` aufgerufen und gibt der Benutzer die folgenden Zeilen, gefolgt von einer Leerzeile, ein,

```
Hello
5
World
1 Line to go
9
```

so soll das Ergebnis `2` sein.

Aufgabe H10.2 Join (7 Punkte)

Die Tutoren schreiben die Übungspunkte in einfache Textdateien (*Übungsdateien*) der folgenden Form:

```
carolus.magnus@rwth-aachen.de,5,5,5,5
maria.stuart@my-head-is-my-own.gov.uk,5,3,4
veni.vidi.bibi@haskell.tum.de,1,2,3,4
```

In jeder Zeile steht zunächst die E-Mail-Adresse und dann die Punktzahlen für die Hausaufgaben einer Woche. Die Einträge sind jeweils durch ein `','` getrennt und die Anzahl der Punktzahlen kann in jeder Zeile variieren.

Zusätzlich gibt es noch eine Liste der E-Mail-Adressen und Namen aller Studenten (*Namensdatei*):

```
maria.stuart@my-head-is-my-own.gov.uk,Maria Stuart
roi.soleil@etat.moi,Louis XIV
veni.vidi.bibi@haskell.tum.de,Julius Caesar
```

Die Übungsleitung braucht jetzt eine Datei, in der sowohl der Name als auch die Übungspunkte enthalten sind (*Join*). Für die obigen Dateien wäre das Ergebnis:

```
maria.stuart@my-head-is-my-own.gov.uk,Maria Stuart,5,3,4
veni.vidi.bibi@haskell.tum.de,Julius Caesar,1,2,3,4
```

Louis XIV fehlt in dem Join, da für ihn keine Punkte eingetragen wurden. Umgekehrt fehlt caroluſ.magnuſ@rwth-aachen.de, da diese Adresse in der Namensdatei nicht vorkommt.

1. Schreiben Sie eine Funktion

   ```
   joinFiles :: FilePath -> FilePath -> IO [String]
   ```

   die den Pfad zu einer Namensdatei und den Pfad zu einer Übungsdatei als Eingabe nimmt und den Join zurückgibt (als Liste von Zeilen). Sie dürfen annehmen, dass die Zeilen einer Datei aufsteigend nach der E-Mail-Adresse sortiert sind und dass jede E-Mail-Adresse höchstens einmal vorkommt.

   Zum Lesen von Dateien können Sie die IO-Aktion `readFile :: FilePath -> IO String` verwenden (`FilePath` ist ein anderer Name für `String`).

2. Wenn Sie ein kompiliertes Haskell-Programm aufrufen, wird die (von Ihnen zu schreibende) IO-Aktion `main :: IO ()` ausgeführt. Nutzen Sie dies, um ein Programm zu schreiben, dass als Kommandozeilenparameter die Namen von Namensdatei und Übungsdatei bekommt und den Join auf dem Bildschirm ausgibt.

   Mittels der IO-Aktion `getArgs` (aus `System.Environment`) können Sie auf die Kommandozeilenparameter zugreifen.

   Zum Kompilieren öffnen Sie eine Kommandozeile und wechseln in das Verzeichnis mit Ihrer Lösung. Mit `ghc -main-is Exercise_10 Exercise_10.hs` kompilieren Sie Ihr Programm in eine ausführbare Datei `Exercise_10` (`Exercise_10.exe` unter Windows). Testen Sie Ihr Programm auf den beigelegten Beispieldateien (der Join von `missing.txt` und `names.txt` soll `grades.txt` sein).

Aufgabe H10.3 Protokollarisch (5 Punkte)

In dieser Aufgabe implementieren Sie einen Netzwerkclient, der sich mit unserem `fp.in.tum.de`-Server verbindet und eine kleine Rechenaufgabe löst.

Die Kommunikation mit dem Server verläuft streng nach dem I2B10H3P (Info 2, Blatt 10, Hausaufgabe 3-Protokoll):

1. Der Client verbindet sich mit dem Host `fp.in.tum.de` auf dem Port `8080`.
2. Bei Verbindungsaufbau meldet sich der Server mit `I2B10H3P/1.0 HELLO`, gefolgt von Whitespaces.
3. Ihr Client soll daraufhin mit Ihrer E-Mail-Adresse (die auch für den Login auf `fp.in.tum.de` genutzt wird), gefolgt von einem Leerzeichen einer beliebigen Zahl n (eine positive Ganzzahl $< 10^3$) und einem Zeilenumbruch antworten.

   Beispiel: `"veni.vidi.bibi@haskell.tum.de 11\n"`

4. Der Server sendet zwei Ganzzahlen a und b im Bereich $0 < a, b < 10^3$, getrennt durch ein Leerzeichen und gefolgt von Whitespaces.
5. Abschließend senden Sie dem Server das Ergebnis der Berechnung $n^a - b$, gefolgt von einem Zeilenumbruch. Optional können Sie dem Server auch eine beliebige Nachricht (in der kein Zeilenumbruch vorkommt) senden, in der an beliebiger Stelle das Ergebnis der Berechnung steht.

   Beispiel 1: `"700\n"`

   Beispiel 2: `"The Next 700 Theorem Provers\n"`

   Beispiel (nicht gültig): `"The Next\n700 Programming Languages\n"`
6. Der Server quittiert den Erhalt der Nachricht und beendet die Verbindung. War diese korrekt, so wird mit `OK`, andernfalls mit `NOT OK`, jeweils gefolgt von Whitespaces, geantwortet.

Der Server wartet jeweils höchstens eine Sekunde auf eine Eingabe Ihres Programms.

Für die volle Punktzahl muss Ihr Programm so implementiert sein, dass ein sich an das Protokoll haltender Server stets mit `OK` antwortet.

Definieren Sie eine Funktion `client :: IO ()`, die einen Client startet und das Protokoll initiiert.

Aufgabe H10.4 Dealing with state (Wettbewerbsaufgabe, 5 Punkte)

> *Das Heil der Demokratien, von welchem Typus und Rang sie immer seien, hängt von einer geringfügigen technischen Einzelheit ab: vom Wahlrecht. Alles andere ist sekundär.*
>
> — José Ortega y Gasset

Für die Wahl zum deutschen Bundestag[1] wird das Areal der Bundesrepublik in Wahlkreise aufgeteilt. Diese Wahlkreise sollen eine möglichst gleichmäßige Einwohnerzahl aufweisen. Die Zuschneidung von Wahlkreisen soll außerdem politisch neutral ablaufen, damit Gerrymandering vermieden wird. In dieser Aufgabe sollen Sie für ein gegebenes Areal (für den Wettbewerb: möglichst gute) Wahlkreise ermitteln.

Gegeben seien folgende Typdefinitionen:

```
type Point = (Rational, Rational)
type Polygon = [Point]
data City = City String Point Integer
data District = District Polygon [City]
```

sowie folgende Annahmen über diese Typen:

- Ein `Point` hat stets nicht-negative Koordinaten.

[1] Der Master of Competition interessiert sich offenbar für Wahlrecht.

- Ein `Polygon` besteht aus mindestens drei Punkten und ist stets konvex. Alle Punkte in einem `Polygon` sind voneinander verschieden. (Das Polygon `[(0, 0), (1, 1), (1, 0)]` steht folglich für ein Dreieck.)

  Hinweis: Im Sinne dieser Aufgabe muss das Polygon keine bestimmte Orientierung haben. Insofern sind sowohl `[(0, 0), (1, 1), (1, 0)]` als auch `[(1, 0), (0, 0), (1, 1)]` gültige Polygone.
- Eine Stadt (`City`) hat einen Namen (`String`), liegt auf einem `Point` und hat eine festgelegte Einwohnerzahl (`Integer`). Die Einwohnerzahl ist positiv (> 0). Städte dürfen auf einer Kante oder einer Ecke des Distrikts liegen.
- Ein Distrikt (`District`) hat eine Grenze (`Polygon`) und Städte, die *innerhalb dieser Grenzen* liegen (`[City]`).

Gesucht wird nun eine Funktion

```
split :: District -> Integer -> [Polygon]
```

so dass der Aufruf `split` d n eine Zerlegung des Distrikts d in n (konvexe, aus mindestens drei Punkten bestehende) Polygone liefert. Sie dürfen annehmen, dass $|d| \geq n + 2$ ist, wobei $|d|$ für die Anzahl der Eckpunkte der Grenzen von d steht. (Anders gesprochen: Eine Aufteilung von d in n Polygone ist möglich.)

Für die berechnete Zerlegung muss zusätzlich folgendes gelten:

1. Die Polygone dürfen sich nicht überlappen; höchstens an Ecken und Kanten berühren.
2. Zusammengesetzt müssen die Polygone den Grenzen des gesamten Distrikts entsprechen.

Für die Hausaufgabe brauchen die Städte nicht weiter beachtet werden. Es reicht aus, dass *irgendeine* gültige Zerlegung in n Polygone gefunden wird.

Für den Wettbewerb hingegen zählt die Verteilung der Einwohnerzahlen innerhalb der Zerlegung, die möglichst gleichmäßig sein soll. Liegt eine Stadt auf einer gemeinsamen Ecke oder Kante zweier oder mehrerer Polygone der Zerlegung, so wird die Stadt dem in der Liste als erstes vorkommenden Polygon zugerechnet. (Daraus folgt auch, dass Städte nicht geteilt werden.)

Ferner darf die Bedingung 2 wie folgt relaxiert werden: Zusammengesetzt müssen die Polygone so innerhalb der Grenzen des gesamten Distrikts liegen, dass alle Städte, die im Distrikt liegen, in mindestens einem Polygon der Zerlegung liegen. Beachten Sie, dass alle Polygone weiterhin konvex sein müssen.

Die Lösung muss innerhalb der Kommentare {`-WETT-`} und {`-TTEW-`} abgegeben werden, um als Wettbewerbsbeitrag zu zählen.

Nur Lösungen, die alle Tests in der Testreihe des MCs bestanden haben, können Wettbewerbspunkte erzielen. Die Richtigkeit ist diese Woche *besonders wichtig*, da der MC von Dr. Angela Merkel höchstpersönlich für die Neuzuschneidung der Wahlkreise in Vorbereitung der nächsten Wahlen beauftragt worden ist. Wie letzte Woche hätten hier Programmierfehler wieder ungeheure Konzequenzen für Deutschland, Europa und die Welt.

Wichtig: Wenn Sie diese Aufgabe als Wettbewerbsaufgabe abgeben, stimmen Sie zu, dass Ihr Name ggf. auf der Ergebnisliste auf unserer Internetseite veröffentlicht wird. Sie können diese Einwilligung jederzeit widerrufen, indem Sie eine Email an `fp@fp.in.tum.de` schicken. Wenn Sie nicht am Wettbewerb teilnehmen, sondern die Aufgabe allein im Rahmen der Hausaufgabe abgeben möchten, lassen Sie bitte die {`-WETT-`} . . . {`-TTEW-`} Kommentare weg. Bei der Bewertung Ihrer Hausaufgabe entsteht Ihnen hierdurch kein Nachteil.