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

WS 2013/14
15.10.2013
Abgabe: 22.10.2013, 15:30

Einführung in die Informatik 2
1. Übung

Aufgabe G1.1 Auswertung

1. Definieren Sie eine Funktion

   ```
   threeDifferent :: Integer -> Integer -> Integer -> Bool
   ```

   die genau dann `True` zurückgibt, wenn alle Parameter unterschiedlich sind.
2. Definieren Sie eine Funktion

   ```
   fourEqual :: Integer -> Integer -> Integer -> Integer -> Bool
   ```

   die genau dann `True` zurückgibt, wenn alle Parameter gleich sind.
3. Geben Sie zu den beiden folgenden Ausdrücken je eine zeilenweise Auswertung an:

   ```
   threeDifferent (2+3) 5 (11 `div` 2)
   fourEqual (2+3) 5 (11 `div` 2) (21 `mod` 11)
   ```

Aufgabe G1.2 This one goes to 11

1. Schreiben Sie eine rekursive Funktion `fac :: Integer -> Integer`, die die Fakultätsfunktion berechnet.
2. Schreiben Sie eine Funktion `sum_eleven :: Integer -> Integer`, die für den Parameter n die Summe $\sum_{i=n}^{n+10} i$ berechnet.

   Hinweis: Gegebenenfalls ist eine Hilfsfunktion nützlich.

Aufgabe G1.3 Maximum

Gegeben sei eine Funktion `g` mit

```
g :: Integer -> Integer
g n = if n < 10 then n*n else n
```

1. Definieren Sie eine rekursive Funktion `max_g :: Integer -> Integer`, so dass `max_g n` die Zahl $0 \leq$ `i` $\leq$ `n` ausgibt, für die `g i` am größten ist. Schreiben Sie diese Funktion so, als ob Sie nichts über `g` wüssten.
2. Schauen Sie sich die Funktion `g` an und überlegen Sie, wann `max_g n` $\neq$ `n` gilt. Verwenden Sie dies, um einen QuickCheck-Test für `max_g` zu schreiben.

Aufgabe H1.1 Oans zwoa vier gsuffa (6 Punkte, Wettbewerbsaufgabe)

Diese Aufgabe ist vom Master of Competition gestellt. Sie wird als Hausaufgabe korrigiert, zählt aber zusätzlich als Teil des informalen Wettbewerbs.[1]

Implementieren Sie eine Funktion `f124 :: Integer -> Integer -> Integer -> Integer`, die drei Argumente x, y, z bekommt und $a + 2b + 4c$ berechnet, wobei a das kleinste, b das zweitkleinste und c das größte Element von x, y, z bezeichnet. (Falls ein Element mehrmals in x, y, z vorkommt, kommt es auch mehrmals in a, b, c vor.)

Beispiele:

```
  f124 0 0 0 == 0          -- 0 == 0 + 2 * 0 + 4 * 0
  f124 0 0 1 == 4          -- 4 == 0 + 2 * 0 + 4 * 1
  f124 1 0 0 == 4
f124 100 1 1 == 403        -- 403 == 1 + 2 * 1 + 4 * 100
f124 1 100 1 == 403
f124 1 10 100 == 421       -- 421 == 1 + 2 * 10 + 4 * 100
f124 10 1 100 == 421
f124 100 1 10 == 421
```

Im Wettbewerb gewinnt die kürzeste Lösung im Hinblick auf die Anzahl der Token. Token sind Identifier, Operatoren, Klammern, Zahlen usw.[2] Die Länge der Identifier ist unerheblich. Der offizielle Tokenzähler ist auf der Vorlesungswebseite erhältlich.[3] Alle Funktionen aus der Haskell-Standardbibliothek sowie eigene Hilfsfunktionen sind erlaubt. Die vollständige Lösung (außer Standardfunktionen) muss innerhalb der Kommentare {-WETT-} und {-TTEW-} abgegeben werden, um als Wettbewerbsbeitrag zu zählen. Zum Beispiel:

```
{-WETT-}
f124 :: Integer -> Integer -> Integer -> Integer
f124 x y z = ...
{-TTEW-}
```

Der Master of Competition hat für seine Lösung 31 Token (exklusive Typsignaturen) gebraucht. Unterbieten Sie ihn!

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.

[1] `http://www21.in.tum.de/teaching/info2/WS1314/wettbewerb.html`
[2] `http://www21.in.tum.de/teaching/info2/WS1314/wettbewerb.html#token`
[3] `http://www21.in.tum.de/teaching/info2/WS1314/Tokenize.hs`

Aufgabe H1.2 Rekursion (8 Punkte)

Sei `f :: Integer -> Integer` wie folgt definiert:

$$\texttt{f}\ n = \begin{cases} n - 10 & \text{if } n > 100 \\ \texttt{f}\ (\texttt{f}\ (n + 11)) & \text{sonst} \end{cases}$$

1. Übertragen Sie die Funktionsdefinition nach Haskell.
2. Evaluieren Sie Ihre in 1 definierte Funktion mit den Parametern $0, 1, 2, 3, 42, 101, 1001, -10$.
3. Geben Sie, basierend auf Ihren Beobachtungen in 2, eine alternative Definition von `f` an und überprüfen Sie Ihre Vermutung mit QuickCheck.

Aufgabe H1.3 Dualer Logarithmus (6 Punkte)

Der duale Logarithmus einer positiven Ganzzahl m ist definiert als die größte Ganzzahl k mit $2^k \leq m$. Implementieren Sie eine Funktion `ld :: Integer -> Integer`, die den dualen Logarithmus berechnet.

Hinweis: Dividieren Sie wiederholt durch 2.