Philipps Universität Marburg

Übungen zu „Konzepte von Programmiersprachen“, WS 2010/11

Prof. Dr. R. Loogen · Fachbereich Mathematik und Informatik · Marburg

Nr. 2, Abgabe: Dienstag, 2. November 2010 vor der Vorlesung

Die Lösungen müssen schriftlich, Programme zusätzlich per E-Mail an den Tutor oder die Tutorin abgegeben werden. Die Abgabe ist in Gruppen bis zu zwei Personen erlaubt.

3. Potenzierung — 4 Punkte

(a) Definieren Sie eine Funktion

```
power :: (Num a, Integral b) => a -> b -> a
```

/ 1

so dass `(power k n)` den Wert k^n liefert. Gehen Sie davon aus, dass $n \geq 0$ gilt. Der vordefinierte Operator `(^) :: (Num a, Integral b) => a -> b -> a` darf nicht verwendet werden.

(b) Geben Sie eine ausführliche Auswertung des Ausdrucks `(power 2 3)` an. / 1

(c) Geben Sie eine alternative Definition der Funktion `power` an, die folgende Gleichungen ausnutzt: / 2

$$k^{2n} = (k^n)^2$$
$$k^{2n+1} = k * (k^n)^2$$

4. Programmierstil — 3 Punkte

Die Funktion `almostEqual :: (Int, Int) -> (Int, Int) -> Bool` vergleicht die Werte von zwei Wertepaaren des Typs `Int` miteinander. Sie liefert das Ergebnis `True`, falls beide Paare dieselben Werte enthalten, wobei deren Reihenfolge nicht von Belang ist. Beispielsweise gilt:

```
almostEqual (3,4) (4,3)  liefert  True
almostEqual (3,4) (3,5)  liefert  False
```

Welche der folgenden Definitionen geben den korrekten Wert zurück? / 1

Welche Definitionen halten Sie für einen guten Programmierstil? Warum? / 1

Fügen Sie den Definitionen Kommentare hinzu, damit diese verständlicher und leichter lesbar werden. / 1

```
almostEqual1 (x1,y1) (x2,y2)
  | (x1 == x2) && (y1 == y2) = True
  | (x1 == y2) && (y1 == x2) = True
  | otherwise                = False

almostEqual2 (x1,y1) (x2,y2)
  | (x1 == x2) = (y1 == y2)
  | (x1 == y2) = (y1 == x2)
  | otherwise  = False

almostEqual3 pair1 pair2
  = (pair1 == pair2) ||
    (pair1 == swap pair2)
  where swap (x,y) = (y,x)

almostEqual4 pair1 pair2
  = (pair1 == pair2) ||
    (swap pair1 == swap pair2)
  where swap (x,y) = (y,x)

almostEqual5 (x1,y1) (x2,y2)
  = if (x1 == x2)
    then if (y1 == y2)
         then True
         else False
    else if (x1 == y2)
         then if (x2 == y1)
              then True
              else False
         else False
```

5. Zeichenkettenverarbeitung 5 Punkte

Definieren Sie die folgenden Funktionen über Zeichenketten, die in Haskell als Listen von Zeichen realisiert sind: `type String = [Char]`.

(a) `delete :: String -> Int -> String` / 1

entfernt beim Aufruf `(delete cs i)` das `i`-te Zeichen von `cs`.

(b) `insert :: String -> Int -> String -> String` / 1

fügt beim Aufruf `(insert cs1 i cs2)` in `cs1` ab der Position `i` den Text `cs2` ein.

(c) `slice :: String -> Int -> Int -> String` / 1

bestimmt beim Aufruf `(slice cs i n)` den Teiltext von `cs`, der an Position `i` beginnt und `n` Zeichen umfasst.

(d) `count :: Char -> String -> Int` / 2

bestimmt beim Aufruf `(count c cs)` die Anzahl der Vorkommen des Buchstabens `c` in der Zeichenkette `cs`.

Ist Ihre Funktion endrekursiv? Entwickeln Sie ggfs. eine endrekursive Definition von `count`.