Algorithmen und Programmierung 1
(Funktionale Programmierung), WS 2011/2012 — Klausur

Aufgabe	1	2	3	4	5	6
Punkte	20	15	20	20	5	10

gesamt: 90

90 Minuten. Abgabe bis Montag, 20. Februar 2012, 15:30 Uhr

---

1. (20 Punkte) Schreiben Sie ein Modul `Stapel` zur Implementierung des abstrakten Datentyps `ST a`, der einen *Stapel* (Keller, *stack*, *pushdown storage*) von Werten vom Typ `a` darstellt. Der Stapel soll folgenden Operationen erlauben:

   ```
   leer:: ST a -- erzeugt einen leeren Stapel
   push:: a -> ST a -> ST a -- fügt ein Element als oberstes ein
   top:: ST a -> a -- greift auf das oberste Element zu
   pop:: ST a -> ST a -- löscht das oberste Element vom Stapel
   istLeer:: ST a -> Bool -- testet, ob der Stapel leer ist
   ```

   Die Bedeutung der Operationen ist durch folgende Gleichungen gegeben.

   ```
   istLeer (leer) = True
   istLeer (push x s) = False
   top (push x s) = x
   pop (push x s) = s
   push (top s) (pop s) = s, falls not (istLeer s)
   ```

   Bei einem leeren Stapel (`istLeer` liefert `True`) soll `top` und `pop` eine Fehlermeldung ausgeben.

   Sie können frei entscheiden, welche Datenstruktur Sie zur Darstellung des Stapels verwenden und wie Sie die Operationen implementieren. Das Modul soll eine explizite Exportliste haben. Ist es erforderlich, `a` auf bestimmte Typklassen einzuschränken? (Diese müssen Sie eventuell bei den Signaturen der obigen Funktionen als Voraussetzungen ergänzen.) Begründen Sie Ihre Antwort.

   Bei allen Funktionen ist wie immer eine Typdeklaration anzugeben.

2. (15 Punkte) Welche Variablen kommen im Ausdruck

   $$\lambda x.\Big(\big(\lambda x.\lambda a.(ax(xx)yx)\big)(\lambda a.xa)\Big)$$

   gebunden vor? Welche Variablen kommen frei vor (bezogen auf den gesamten Ausdruck)?

   Reduzieren Sie den Ausdruck so weit wie möglich.

3. (20 Punkte) Summen

   Schreiben Sie ein Haskell-Programm, das in einer Liste vom Typ `[Integer]` ein Tripel (a, b, c) mit $a+b = c$ findet, falls es eines gibt. Dabei müssen a, b und c von *verschiedenen* Stellen der Eingabeliste kommen. (Sie dürfen gleich sein, falls in der Eingabeliste gleiche Zahlen mehrfach vorkommen.) Sie dürfen die Funktion `sort` aus dem Modul `Data.List` verwenden, die eine Folge der Länge n mit Laufzeit $O(n \log n)$ sortiert. Verwenden Sie Kommentare zur Erläuterung Ihres Programms. Bei allen Funktionen ist wie immer eine Typdeklaration anzugeben.

   Analysieren Sie sie asymptotische Laufzeit Ihres Programms (in O-Notation in Abhängigkeit von der Länge n der Eingabeliste).

   (Falls Ihr Programm korrekt ist und eine bessere Laufzeit als $O(n^3)$ hat, gibt es bis zu 10 Zusatzpunkte.)

4. (20 Punkte) Welche Formel muss man für x einsetzen, damit die Gleichung

   `drop m . take n = take (`x`) . drop m`

   für alle $m \geq 0$ und alle $n \geq 0$ gilt?

   Beweisen Sie die Gleichung dann durch strukturelle Induktion (oder durch eine andere Methode Ihrer Wahl) unter Verwendung der folgenden Definitionen:

```
     take, drop:: Int -> [a] -> [a]
(t1) take n _  | n <= 0  = []
(t2) take _ []           = []
(t3) take n (x:xs)       = x : take (n-1) xs
(d1) drop n xs | n <= 0  = xs
(d2) drop _ []           = []
(d3) drop n (_:xs)       = drop (n-1) xs
```

5. (5 Punkte) Funktionen höherer Ordnung

   Welcher der folgenden Ausdrücke ist äquivalent zur Formel `[g x y | y <- ys]`?

   a) `(map . g x) ys`, b) `(map g x) ys`, c) `map(g x y) ys`, d) `map (g x)ys`.

   (Es können auch mehrere Antworten richtig sein. Eine Begründung ist nicht erforderlich.)

6. (10 Punkte) Typinferenz

   Bestimmen Sie den allgemeinsten Typ, den die Funktion `f` haben kann.

```
f a b c
        | b c       = [c+2]
        | otherwise = a c
```

   Begründen Sie Schritt für Schritt, wie Sie zu Ihren Schlussfolgerungen kommen.