Informatik I

Probeklausur WS 2011/2012

Es sind maximal 60 Punkte erreichbar, 36 Punkte $\widehat{=}$ 60% sind hinreichend zum Bestehen (Note 4,0). Verwenden Sie die Konstruktionsanleitungen!

Hilfsmittel: keine — Bearbeitungszeit: 60 Minuten

Aufgabe 1 $(1 + 1 + 1 + 1 = 4$ Punkte)

Definieren Sie die folgenden Begriffe kurz (maximal 140 Zeichen, also eine Twitter-Nachricht):

a) endrekursive Prozedur

b) promise

c) parametrisch polymorphe Prozedur

d) curry bzw. currying

Aufgabe 2 (3 Punkte)

Entscheiden Sie, welche Scheme-Ausdrücke a–e) welche der Signaturen 1–5 erfüllen:

a)	`empty`	1.	`(%a -> %b)`
b)	`(lambda () "Hello")`	2.	`(promise %a)`
c)	`(list 3 2 1)`	3.	`(-> (%a -> %b))`
d)	`(lambda () (lambda (b) b))`	4.	`(list-of (list-of %a))`
e)	`list-of`	5.	`(pair-of %a empty-list)`

Aufgabe 3 $(2 + 1 + 7 = 10$ Punkte)

Schreiben Sie Daten- und Record-Definitionen für Teller und Tassen. Ein Teller ist charakterisiert durch seinen Durchmesser, eine Tasse durch ihr Volumen. Zusätzlich verfügen Tassen und Teller jeweils über eine Farbe (als String repräsentierbar).

Ein Geschirrstück sei entweder eine Tasse oder ein Teller. Definieren Sie hierfür einen Datentyp und schreiben Sie eine Prozedur, die eine Liste von Geschirrstücken konsumiert und bestimmt, ob alle Bestandteile der Liste dieselbe Farbe haben.

Aufgabe 4 $(2 + 9 = 11$ Punkte)

Verwenden Sie für diese Aufgabe die Ihnen bekannten Definitionen der Prozeduren `fold` und `append`.

a) Geben Sie *möglichst allgemeine* Signaturen von `fold` und `append` an.

b) Beweisen Sie mithilfe von struktureller Induktion, dass für zwei beliebige Listen ℓ_1 und ℓ_2 folgender Zusammenhang gilt:

$$\texttt{(fold } \ell_2 \texttt{ make-pair } \ell_1\texttt{)} \equiv \texttt{(append } \ell_1 \ \ell_2\texttt{)}$$

Aufgabe 5 (3 + 3 = 6 Punkte)

Gegeben ist folgende Prozedur:

```
(define mystery
  (lambda (x y z)
    (if (z y)
        (list y)
        (make-pair y
                   (mystery x (x y) z)))))
```

a) Geben Sie eine *möglichst allgemeine* Signatur von `mystery` an.

b) Schreiben Sie einen Funktionsaufruf von `mystery`, dessen Ergebnis eine Liste der natürlichen Zahlen von 1 bis 10 ist.

Aufgabe 6 (6 Punkte)

Schreiben Sie eine Prozedur `and-filter`, die eine Liste ℓ und eine Liste p aus Prädikaten konsumiert; `and-filter` soll aus ℓ alle Elemente entfernen, die nicht alle Prädikate aus p erfüllen. Schreiben sie *mindestens 4* interessante Testfälle.

Aufgabe 7 (1 + 3 + 3 + 4 = 11 Punkte)

Unendliche Ströme lassen sich nicht durch Listen darstellen und umgekehrt. Beispielsweise kann ein Aufruf von `stream-filter` zu Endlosrekursion führen, wenn kein Element des Streams das entsprechende Prädikat erfüllt. In dieser Aufgabe sollen die Vorteile von Listen und Streams in einem neuen Datentyp vereint werden – potentiell (aber nicht zwangsweise) endlichen Strömen. Gehen Sie von folgender Grundlage aus:

```
; "Versprechen", einen Wert der Signatur t liefern zu können
(define promise
  (lambda (t)
    (signature (-> t))))

; "Einlösung" (Auswertung) des Versprechens p
(: force ((promise %a) -> %a))
(define force
  (lambda (p)
    (p)))

; Polymorphe Paare (isomorph zu 'pair')
(: make-cons (%a %b -> (cons-of %a %b)))
(define-record-procedures-parametric cons cons-of
  make-cons cons?
  (head tail))
```

Hinweis: Sie brauchen für die folgenden Prozeduren weder Testfälle noch Kommentare zu schreiben (Signaturen sind aber erforderlich).

a) Schreiben Sie eine Record-Definition für den leeren Strom (`empty-stream`), der über keine weiteren Eigenschaften verfügt. Dieser markiert das Ende eines Stroms.

b) Schreiben Sie eine Prozedur `stream-of`, die eine Signatur s konsumiert und eine Signatur für potentiell endliche Ströme aus Elementen vom Typ s erzeugt. Ein potentiell endlicher Strom ist also entweder der leere Strom oder ein entsprechender `cons`-Record.

c) Schreiben Sie eine Prozedur `stream-take`, die eine natürliche Zahl $n \in \mathbb{N}_0$ und einen potentiell endlichen Strom konsumiert und dessen erste n Elemente in einer Liste ausgibt (wenn der Strom weniger als n Elemente hat, so sollen alle Elemente des Stroms in einer Liste ausgegeben werden).

d) Schreiben Sie nun eine Prozedur `to`, die eine natürliche Zahl $n \in \mathbb{N}_0$ konsumiert und den endlichen Strom der natürlichen Zahlen $0, \ldots, n$ erzeugt. *Hinweis:* Sie können hierfür eine Hilfsprozedur schreiben, die neben n noch eine Variable c konsumiert, die pro rekursiven Aufruf um 1 erhöht wird und für $n = c$ zum Abbruch der Rekursion führt.

Aufgabe 8 $(2 + 3 + 4 = 9$ Punkte)

Betrachten Sie die folgenden λ-Terme:

i. $((((\lambda x.x)\ (\lambda z.(\lambda y.((z\ y)\ y))))\ (\lambda x.(\lambda y.((y\ y)\ (y\ x)))))\ (\lambda z.z))$

ii. $(((\lambda y.((\lambda x.y)\ ((\lambda x.(x\ x))\ (\lambda y.(y\ y)))))\ (\lambda x.x))\ (\lambda x.(y\ z)))$

iii. $((\lambda z.z)\ (((\lambda y.(\lambda x.(y\ x)))\ (\lambda z.x))\ ((\lambda x.(x\ x))\ (\lambda y.(\lambda z.(y\ z))))))$

a) Unterstreichen Sie in den λ-Termen alle β-Redexe.

b) Welches sind jeweils die freien und welches die gebundenen Variablen der λ-Terme?

c) Geben Sie die Normalformen für den zweiten und den dritten λ-Term an.