Fachschaft Informatikstudiengänge

Fachrichtung 6.2 — Informatik

Das Team der Bremser

1. Probeklausur zu Programmierung 1 (WS 07/08)

http://fsinfo.cs.uni-sb.de

Name Matrikelnummer

Bitte öffnen Sie das Klausurheft erst dann, wenn Sie dazu aufgefordert werden.
Sie können die Klausur nur auf dem für Sie vorgesehen Platz schreiben. Sie müssen das mit Ihrem Namen und Ihrer Matrikelnummer versehene Heft verwenden.
Hilfsmittel sind nicht zugelassen. Am Arbeitsplatz dürfen nur Schreibgeräte, Getränke, Speisen und Ausweise mitgeführt werden. Taschen und Jacken müssen an den Wänden des Klausursaals zurückgelassen werden.
Das Verlassen des Saals ohne Abgabe des Klausurhefts gilt als Täuschungsversuch.
[Wenn Sie während der Bearbeitung zur Toilette müssen, geben Sie bitte Ihr Klausurheft bei der Aufsicht ab. Es kann immer nur eine Person zur Toilette.]
Alle Lösungen müssen auf den bedruckten rechten Seiten des Klausurhefts notiert werden. Die leeren linken Seiten dienen als Platz für Skizzen und werden **nicht korrigiert.** Notizpapier ist nicht zugelassen. Sie können mit Bleistift schreiben.
Für die Bearbeitung der Klausur stehen 75 [150] Minuten zur Verfügung. Insgesamt können 75 [150] Punkte erreicht werden. Die für jede Aufgabe angebene Punktzahl gibt Ihnen also einen Anhaltspunkt, wieviel Zeit Sie auf die Bearbeitung der Aufgabe verwenden sollten. Zum Bestehen der Klausur genügen 37,5 [75] Punkte.
[Bitte legen Sie zur Identifikation Ihren Personalausweis bzw. Reisepass sowie Ihren Studierendenausweis neben sich.]
Viel Erfolg!

1	2	3	4	5
15	15	14	16	15

Summe
75

Note

Deklarationen

Sie können die folgenden vordeklarierten Prozeduren benutzen:

$$
\begin{aligned}
op@ &: \alpha\ list * \alpha\ list \to \alpha\ list \\
map &: (\alpha \to \beta) \to \alpha\ list \to \beta\ list \\
foldl &: (\alpha * \beta \to \beta) \to \beta \to \alpha\ list \to \beta \\
iter &: int \to \alpha \to (\alpha \to \alpha) \to \alpha \\
first &: int \to (int \to bool) \to int \\
List.concat &: \alpha\ list\ list \to \alpha\ list
\end{aligned}
$$

Aufgabe 1.1 (Sortieren, 15=12+3)

a) Schreiben Sie eine polymorphe Prozedur *smergesort* : $(\alpha * \alpha \rightarrow order) \rightarrow \alpha\ list \rightarrow \alpha\ list$, die Listen gemäß einer Vergleichsprozedur mit Sortieren durch Mischen strikt sortiert. Deklarieren Sie auch die dafür verwendeten Prozeduren *split* und *smerge*.

```
datatype order = LESS | EQUAL | GREATER

fun split
```

```
fun smerge compare
```

```
fun smergesort compare
```

b) Wie arbeitet dieses Verfahren (3 Sätze)? Welche Eigenschaften zeichnen dieses Verfahren aus (z.B. in Bezug auf Laufzeit, vorsortierte Listen,...)?

Aufgabe 1.2 (Logische Ausdrücke, 15=7+1+7)

Seien logische Ausdrücke mit Negation, Disjunktion, sowie Konjunktion wie folgt dargestellt:

```
datatype log = TRUE | FALSE | NOT of log | OR of log * log | AND of log * log
```

a) Deklarieren Sie eine Prozedur *simplify* : *log* → *log*, die einen logischen Ausdruck vereinfacht. Zum Beispiel soll $OR(TRUE, FALSE)$ den Wert *TRUE* liefern.

```
fun simplify
```

b) Erweitern Sie den Datentyp um zwei Konstruktoren *ORL* und *ANDL*, die eine Liste von Ausdrücken mit *OR* bzw. *AND* darstellen, also Disjunktion und Konjunktion mit beliebiger Stelligkeit.

```
datatype log = TRUE | ...
             ⋮
             | ORL of
             | ANDL of
```

c) Erweitern Sie nun die Prozedur *simplify* : *log* → *log* um diese beiden Operationen. Beispielsweise soll $ANDL[TRUE, FALSE, TRUE]$ zu *FALSE* und $ORL[TRUE, FALSE, TRUE]$ zu *TRUE* vereinfacht werden. Geben Sie nur die zusätzlichen Regeln an. Wie geht man mit leeren Listen um?

```
|   simplify

|   simplify
```

Aufgabe 1.3 (Endrekursion und Akkus, 14=3+6+5)

Die Fibonacci-Folge sei definiert durch:

$$\begin{aligned} fib(0) &= 0 \\ fib(1) &= 1 \\ fib(n) &= fib(n-1) + fib(n-2) \qquad \text{für } n \geq 2 \end{aligned}$$

a) Schreiben Sie eine Prozedur $\mathit{fib} : \mathit{int} \to \mathit{int}$, die die n-te Fibonacci-Zahl gemäß der gegebenen Gleichungen berechnet.

b) Die Fibonacci-Zahlen lassen sich auch endrekursiv bestimmen. Konstruieren Sie eine Prozedur $\mathit{fibi} : \mathit{int} \to \mathit{int} \to \mathit{int} \to \mathit{int}$, für die $\mathit{fib}\, n = \mathit{fibi}\ 0\ 1\ (n - 1)$ für alle $n \in \mathbb{N}$ gilt. Die ersten beiden Argumente sollen als Akkumulatorargumente dienen.

c) Geben Sie das Ausführungsprotokoll für den Aufruf $\mathit{fibi}\,(0+1)\,(1+1)\,3$ an. Sie sollten 13 Schritte erhalten.

Aufgabe 1.4 (Iteration und Listen, 16=9+7)

a) Deklarieren Sie mit Hilfe von *first* eine Prozedur $double : real \to int$, die zu einem gegebenen Prozentsatz (z.B. 0.04 für 4 Prozent) die Anzahl der Jahre liefert, die es braucht, bis sich das Geld mindestens verdoppelt hat. Deklarieren Sie dafür zuerst unter Verwendung von *iter* eine Prozedur $power : real \to int \to real$, die die n-te Potenz einer Gleitkommazahl berechnet.

```
fun power a n =
```

```
fun double x =
```

b) Deklarieren Sie eine Prozedur $kar : \alpha\ list \to \beta\ list \to (\alpha * \beta)\ list$, die das kartesische Produkt zweier Listen berechnet. Beispielsweise soll $kar[1,2][3,4]$ die Liste $[(1,3),(1,4),(2,3),(2,4)]$ liefern. Auf die Reihenfolge der Tupel kommt es dabei nicht an.

```
fun kar
```

Aufgabe 1.5 (Bezeichnerbindung, 15=5+5+5)

a) Geben Sie ein Typschema an, mit dem der Interpreter die Prozedur

```
fun f a b c d = a = b c = d
```

typisieren wird. Achten Sie auf minimale Klammerung.

b) Bereinigen Sie das folgende Programm durch Indizieren der Bezeichnerauftreten und Überstreichen der definierenden Auftreten.

```
exception E of int

val ( x , y , z ) = ( (), 2 , Empty )

fun f x = ( x ; raise E y )

val x = f x handle y => y

val q = x
```

c) Woran wird q nach Ausführung des Programms gebunden? Welchen Typ hat q?