13. Übungsblatt zu Programmierung WS 2002 / 03

Prof. Dr. Gert Smolka und Dipl.-Inform. Thorsten Brunklaus
www.ps.uni-sb.de/courses/prog-ws02/

Abgabe: Montag, 3. Februar 2003

Schicken Sie Ihre Lösungen per E-Mail an Ihren Übungsgruppenleiter. Geben Sie in der E-Mail zuerst ihren Namen und die Nummer Ihrer Übungsgruppe an. Bitte reichen Sie alle Lösungen mit nur einer E-Mail ein.

Aufgabe 13.1: Referenzen (2) Zu welchem Wert evaluiert der Ausdruck

```
ref 1 = ref 1
```

Begründen Sie ihre Aussage.

Aufgabe 13.2: Generatoren ($15 = 3 + 3 + 3 + 6$) Ein Generator für eine Folge $x_1, x_2, \ldots$ von Werten eines Typs t ist eine Prozedur `unit` $\to t$, die beim n-ten Aufruf x_n liefert.

(a) Schreiben Sie einen Generator `nextSquare` für die Folge 0, 1, 4, 9, . . . der Quadratzahlen.

(b) Schreiben Sie eine Prozedur

```
newNextSquare : unit → unit → int
```

die bei jedem Aufruf einen neuen Generator für die Folge der Quadratzahlen liefert.

(c) Schreiben Sie eine Prozedur

```
newGenerator : (int → 'a) → unit → 'a
```

die zu einer Prozedur f einen Generator für die Folge

$$f(0),\ f(1),\ f(2),\ \ldots$$

liefert.

(d) Schreiben Sie eine Prozedur

```
newNextPrime : unit → unit → int
```

die bei jedem Aufruf einen neuen Generator für die Folge $2, 3, 5, 7, \ldots$ der Primzahlen liefert.

Aufgabe 13.3: Effiziente imperative Schlangen (15) Schreiben Sie eine Struktur, die imperative Schlangen gemäß der folgenden Spezifikation realisiert:

```
type 'a queue
val queue  : unit → 'a queue
val insert : 'a * 'a queue → unit
val head   : 'a queue → 'a          (* Empty *)
val remove : 'a queue → unit        (* Empty *)
val empty  : 'a queue → bool
```

Die Operation `queue` liefert eine neue Schlange, die noch keine Einträge enthält. Die Operation `insert` trägt einen Wert in eine Schlange ein. Die Operation `head` liefert den ältesten Eintrag in einer Schlange. Die Operation `remove` entfernt den ältesten Eintrag aus einer Schlange. Die Operation `empty` testet, ob eine Schlange Einträge enthält.
Beachten Sie unbedingt die folgenden Hinweise:

(a) Alle Operationen sollen konstante Laufzeit haben.

(b) Stellen Sie die Einträge in der Schlange in imperativen Listen gemäß der folgenden Typdeklaration dar:

```
datatype 'a ilist = Nil | Cons of 'a * 'a ilist ref
```

(c) Stellen Sie eine imperative Schlange durch ein Paar

(a, b) `: 'a ilist ref * 'a ilist ref ref`

dar. Dabei soll a auf die imperative Liste xs der Einträge zeigen. Wenn xs nicht leer ist, soll b auf die letzte Referenz von xs zeigen (damit `insert` mit konstanter Laufzeit realisiert werden kann). Wenn xs leer ist, soll b auf a zeigen.

Aufgabe 13.4: Reversieren von Reihungen (6) Schreiben Sie eine Prozedur

```
reverse : 'a array → unit
```

die eine Reihung reversiert. Die Prozedur soll mit einer Schleife und ohne Rekursion realisiert werden. Verwenden Sie die Prozedur `swap` aus dem Skript.

Aufgabe 13.5: Rotieren von Reihungen $(5 + 5)$ Sie sollen eine Prozedur

```
rotate : 'a array → unit
```

schreiben, die die Komponenten einer nichtleeren Reihung um eine Position nach rechts schiebt. Dabei soll die letzte Komponente an die Stelle der ersten Komponente rücken.

(a) Schreiben Sie `rotate` mithilfe einer iterativ rekursiven Hilfsprozedur `rotate'`.

(b) Schreiben Sie `rotate` mithilfe einer Schleife.

Aufgabe 13.6: Binäre Suche in Reihungen (9) Eine Reihung des Types `int array` heißt sortiert, wenn ihre Komponenten aufsteigend sortiert sind. Schreiben Sie eine Prozedur

```
member : int array → int → bool
```

die zu einer sortierten Reihung a und einer Zahl x entscheidet, ob eine Komponente von a den Wert x hat. Für Reihungen der Länge n soll `member` die Laufzeit $O(\log n)$ haben.
Die logarithmische Laufzeit läßt sich mithilfe von binäre Suche erreichen, eine Technik, die wir bereits im Zusammenhang mit Rot-Schwarz-Bäumen kennengelernt haben. Dazu stellen wir uns die sortierte Reihung als eine Folge vor, die von links nach rechts läuft und gehen wie folgt vor:

(a) Bestimme einen Index m, der etwa in der Mitte der Reihung liegt.

(b) Wenn x der Wert der m-ten Komponente ist, liefere `true`.

(c) Wenn x kleiner als der Wert der m-ten Komponente ist, suche in der linken Teilreihung weiter.

(d) Wenn x größer als der Wert der m-ten Komponente ist, suche in der rechten Teilreihung weiter.

(e) Wenn die zu durchsuchende Teilreihung leer ist, liefere `false`.

Eine Teilreihung können wir durch das Paar aus ihrem kleinsten und größten Index darstellen.

Aufgabe 13.7: Reversieren von imperativen Listen (5) Schreiben Sie eine Prozedur

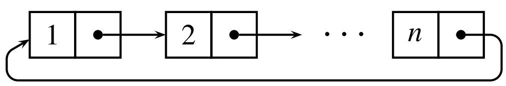
```
reverse : 'a ilist → 'a ilist
```

die eine imperative Liste reversiert indem sie die Zeiger umsetzt und den letzten Kasten liefert. Die Prozedur soll keine neuen Referenzen einführen. Für zyklische Listen soll die Prozedur divergieren.

Aufgabe 13.8: Zyklische imperative Listen (8) Schreiben Sie eine Prozedur

```
circle : int → int ilist
```

die zu $n \geq 1$ eine zyklische Liste mit n Knoten liefert, die mit den Zahlen $1, \ldots, n$ markiert sind:



Die Prozedur soll den mit 1 markierten Knoten liefern.

Aufgabe 13.9: Größe von imperativen Listen (10) Schreiben Sie eine Prozedur

```
size : 'a ilist → int
```

die die Größe einer imperativen Liste liefert (das ist die Anzahl der Referenzen der Liste). Schreiben Sie dazu zuerst eine Prozedur

```
reflist : 'a ilist → 'a ilist ref list
```

die die Liste aller Referenzen einer imperativen Liste liefert. Beide Prozeduren sollen auch für zyklische Listen funktionieren. Geben Sie die Laufzeit Ihrer Prozedur `size` an.

Aufgabe 13.10: Listenreversion mit Schleifen (5) Schreiben Sie mithilfe einer Schleife eine nicht-rekursive Prozedur

reverse : 'a list → 'a list

die Listen mit linearer Laufzeit reversiert.

Aufgabe 13.11: Statische Semantik von F mit Referenzen (6) Sei F wie in Skript gezeigt um Referenzen erweitert. In Kapitel 11 wurde die statische Semantik von F mithilfe von Inferenzregeln definiert. Geben Sie die für Referenzen zusätzlich erforderlichen Regeln an. (Keine Abgabe: die Punkte werden Ihnen ohne Abgabe zuerkannt.)

Aufgabe 13.12: Fakultät in F plus Referenzen (9) Schreiben Sie in dem um Referenzen erweiterten F einen zulässigen Ausdruck, der zu einer Prozedur evaluiert, die die Fakultätsfunktion berechnet.