Christian-Albrechts-Universität zu Kiel
Institut für Informatik
Lehrstuhl für Programmiersprachen und Übersetzerkonstruktion
Prof. Dr. M. Hanus, F. Reck

Institut für Informatik

10. Übung zur Vorlesung „Informatik I“

Abgabe am Freitag, 16. Januar, 14.00 Uhr

Hinweis: Die Übungen beginnen ab dem 12. Januar 2009. Die Vorlesung findet bereits ab dem 9. Januar 2009 wieder statt.
Hinweis: Ab dieser Serie müssen Sie in DrScheme den Sprachlevel *Advanced Student* einstellen, um Zuweisungen und Prozeduren ohne Parameter verwenden zu können.

Präsenzaufgabe 1

In der Vorlesung wurde das Umgebungsmodell zur Formalisierung lokaler Definitionen vorgestellt. Zeichnen Sie die Umgebungen, Bindungen und Prozeduren, die bei der Auswertung von

```
(define (setfirst! l x)
  (local ((define y (first l))) (set! y x)))

(define (one23)
  (local ((define l (list 1 2 3)))
    (begin (setfirst! l 4) l)))
```

entstehen. Erweitern Sie das Diagramm um die Objekte, die durch die Auswertung von `(one23)` entstehen. Was ist der Rückgabewert dieses Aufrufs?

Aufgabe 2 — 6 Punkte

Eine Funktion f heißt *universelle Faltung*, falls es Werte h und e gibt, so dass gilt:

```
(f empty) = e
(f (cons x xs)) = (h x (f xs))
```

Zeigen Sie, dass es zu gegebenen Werten[1] h und e genau eine universelle Faltung gibt.

(a) *Existenz:* Definieren Sie eine Scheme-Funktion `(unifalt h e l)`, so dass

```
(define (f l) (unifalt h e l))
```

für alle Werte `h` und `e` eine universelle Faltung ist.

(b) *Eindeutigkeit:* Zeigen Sie per struktureller Induktion nach l, dass für zwei universelle Faltungen f und g (zu gegebenen Werten h und e) für alle Listen l gilt: $(f\ l) = (g\ l)$.

[1] h muss eine zweistellige Funktion sein

Aufgabe 3 (Programmieraufgabe) 6 Punkte

Definieren Sie einen Datentyp für Paare und eine Funktion `unzip`, die eine Liste von Paaren in ein Paar von Listen umwandelt. Zum Beispiel soll der Aufruf

```
(unzip (list (make-paar 1 2) (make-paar 3 4) (make-paar 5 6)))
```

zu

```
(make-paar (list 1 3 5) (list 2 4 6))
```

ausgewertet werden. Geben Sie eine Implementierung mit linearer Laufzeit an!

Aufgabe 4 (Programmieraufgabe) 8 Punkte

In dieser Aufgabe sollen Sie die Technik der *Nachrichtenweitergabe* anwenden um ein Feld variabler Größe zu implementieren. Ein Feld speichert beliebige Elemente an von 1 an aufwärts nummerierten Positionen und soll vier Operationen zur Verfügung stellen:

- `(size feld)` soll die aktuelle Größe des Feldes `feld` zurückliefern.
- `(get feld n)` soll das `n`-te Element des Feldes `feld` zurückliefern (und einen Fehler liefern, wenn `n` größer ist als die aktuelle Größe des Feldes).
- `(set feld n x)` soll das Feld `feld` so verändern, dass an Position `n` das Element `x` steht und alle anderen Positionen unverändert bleiben. Auch diese Funktion soll einen Fehler liefern, wenn `n` größer ist als die aktuelle Größe des Feldes.
- `(grow feld n)` soll das Feld `feld` um `n` Elemente, die mit `void` initialisiert werden, erweitern.

Definieren Sie eine Funktion `(konstr-feld n)`, die ein `n`-elementiges Feld zurückgibt, das als internen Zustand zum Beispiel eine Liste speichert, deren Elemente mit `void` initialisiert werden.