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

Institut für Informatik

6. Übung zur Vorlesung „Informatik I“

Abgabe am Freitag, 5. Dezember, 14.00 Uhr

Programmierpraktikum: Bitte beachten Sie, dass am Mittwoch den 3. Dezember um 14.15 Uhr die nächste Vorlesung zum Programmierpraktikum stattfindet.
Hinweis: Von dieser Serie an müssen Sie in DrScheme den Sprachlevel *Intermediate Student with lambda* einstellen, damit Funktionen höherer Ordung und lokale Deklarationen unterstützt werden.

Präsenzaufgabe 1

(a) Ermitteln Sie in der angegebenen Nachricht die Häufigkeit der einzelnen Buchstaben. Skizzieren Sie den zugehörigen Huffman-Baum und codieren Sie die Nachricht mit dem dadurch beschriebenen Code.

`IN ULM UM ULM UND UM ULM HERUM`

Aus wievielen Bits besteht die codierte Nachricht? Wieviele Bits hätte man mindestens mit einem Code fester Länge benötigt?

(b) Für alle $n \geq 2$ sei $A_n = \{a_1, \ldots, a_n\}$ ein Alphabet aus n Buchstaben. Der Buchstabe a_i habe für alle $i \in \{1, \ldots, n\}$ das Gewicht 2^{i-1}.

Skizzieren Sie den Huffman-Baum für A_5 und A_{10}. Wieviele Bits werden benötigt, um im Baum für A_n den Buchstaben a_i zu codieren?

Präsenzaufgabe 2

Welchen Wert haben folgende Scheme-Ausdrücke?

- `(local ((define f (lambda (x y) (+ (* 2 x) y)))) (f 11 20))`
- `(local ((define x 3)) (+ x (local ((define y (* x x))) (+ y y))))`
- `((lambda (a b c) (c b a)) 8 - (lambda (x y) (x y 1)))`
- `(cons (rest (cons 1 (list 2 3))) (list (cons 4 (list 5))))`

Aufgabe 3 (Programmieraufgabe) 4 Punkte

Definieren Sie eine Funktion `(filter-list ? l)` mit einem Prädikat `?` und einer Liste `l` als Parameter. Die Funktion soll die Liste derjenigen Elemente von `l` berechnen, für die das Prädikat `?` erfüllt ist. Ein Prädikat ist eine einstellige Funktion, die als Ergebnis `true` oder `false` zurückgibt. Zum Beispiel soll der Aufruf

```
(filter-list (lambda (x) (> x 3)) (list 1 4 2 3 6 4 7))
```

zu `(list 4 6 4 7)` ausgewertet werden.

Aufgabe 4 (Programmieraufgabe) 5+1 Punkte

Definieren Sie eine Funktion `(flatten t)`, die einen als geschachtelte Liste dargestellten Baum `t` *flach klopft*, d.h. seine Blätter in einer Liste zurückgibt. Zum Beispiel soll der Aufruf

```
(flatten (list 1 (list 2 (list 3 4) (list 5)) (list (list 6 7))))
```

zu `(list 1 2 3 4 5 6 7)` ausgewertet werden. Definieren Sie dazu ein Prädikat `list?`, das testet, ob sein Argument eine Liste ist. Können Sie bekannte Funktionen wiederverwenden, um `flatten` zu definieren?

Aufgabe 5 (Programmieraufgabe) 5+5 Punkte

In der Vorlesung wurden Huffman-Bäume zur Codierung und Decodierung von Nachrichten vorgestellt. Die in der Vorlesung vorgestellten Programmteile von Huffman-Bäumen können auf der Seite zu den Übungen heruntergeladen werden. **Im iLearn-System brauchen Sie nur die Funktionen abzugeben, die Sie selbst geschrieben haben. Das Programm von der Website wird für den Test automatisch hinzugeladen.**

(a) Die folgende Funktion erzeugt aus einer Liste `sh-paare` von Symbol/Häufigkeits-Paaren einen Huffman-Baum nach dem in der Vorlesung vorgestellten Verfahren:

```
(define (generiere-huffman-baum sh-paare)
  (nacheinander-vereinen (konstr-blatt-menge sh-paare)))

(define (nacheinander-vereinen l) ... )
```

Hierbei ist `konstr-blatt-menge` die in der Vorlesung vorgestellte Funktion, die eine Liste von Paaren in eine geordnete Menge von Blättern umwandelt.

Implementieren Sie die Funktion `nacheinander-vereinen`, die nacheinander jeweils die Elemente der Menge `l` mit der geringsten Wichtung miteinander vereint, bis nur noch ein Element übrig ist, das den generierten Huffman-Baum darstellt.

(b) Die folgende Funktion erzeugt aus einer Nachricht `nachricht` und einem Huffman-Baum `baum` die codierte Nachricht als eine Liste von Bits:

```
(define (codiere baum nachricht)
  (if  (empty? nachricht)
       empty
       (append (codiere-symbol baum (first nachricht))
               (codiere baum (rest nachricht)))))

(define (codiere-symbol baum symbol) ... )
```

Implementieren Sie die Funktion `codiere-symbol`, die als Ergebnis die Codierung von `symbol` gemäß `baum` als eine Liste von Bits liefert. Ist `symbol` nicht in `baum` vorhanden, so soll die Funktion einen Fehler liefern.

Die Funktion `(error l s)` hat ein Label und einen String als Parameter und bricht das laufende Programm mit einer Fehlermeldung ab. Zum Beispiel erzeugt

```
(error 'codiere-symbol "Symbol nicht im Baum vorhanden")
```

die folgende Fehlermeldung:

```
codiere-symbol: Symbol nicht im Baum vorhanden
```