Informatik I – Programmierung:
Implementierung von Huffman-Bäumen

```
;;; Darstellung von Blaettern

(define-struct blatt (symbol wichtung))


;;; Darstellung von Knoten

(define-struct baum (symbole wichtung links rechts))

(define (konstr-code-baum links rechts)
  (make-baum (append (symbole links) (symbole rechts))
             (+ (wichtung links) (wichtung rechts))
             links
             rechts))

(define (symbole baum)
  (if (blatt? baum)
      (list (blatt-symbol baum))
      (baum-symbole baum)))

(define (wichtung baum)
  (if (blatt? baum)
      (blatt-wichtung baum)
      (baum-wichtung baum)))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; Decodieren von Nachrichten

(define (decodiere bits baum)
  (decodiere-1 bits baum baum))

(define (decodiere-1 bits baum aktueller-ast)
  (if (empty? bits)
      empty
      (decodiere-naechsten-ast (rest bits)
                               baum
                               (waehle-ast (first bits) aktueller-ast))))
```

```
(define (decodiere-naechsten-ast restbits baum naechster-ast)
  (if (blatt? naechster-ast)
      (cons (blatt-symbol naechster-ast)
            (decodiere-1 restbits baum baum))
      (decodiere-1 restbits baum naechster-ast)))

(define (waehle-ast bit ast)
  (cond ((= bit 0) (baum-links ast))
        ((= bit 1) (baum-rechts ast))
        (else (error 'waehle-ast "falsches Bit"))))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; Generierung von Huffman-Baeumen

;;; Einfuegen in geordnete Menge von gewichteten Elementen

(define (hinzufuegen-menge x menge)
  (cond ((empty? menge) (list x))
        ((< (wichtung x) (wichtung (first menge))) (cons x menge))
        (else (cons (first menge)
                    (hinzufuegen-menge x (rest menge))))))


;;; Darstellung von Symbol-Haeufigkeit-Paaren

(define-struct sh (symbol haeufigkeit))


;;; Transformiere eine Liste von Symbol-Haeufigkeit-Paaren
;;; in eine gewichtete Menge von Blaettern

(define (konstr-blatt-menge sh-paare)
  (if (empty? sh-paare)
      empty
      (hinzufuegen-menge (make-blatt (sh-symbol (first sh-paare))
                                     (sh-haeufigkeit (first sh-paare)))
                         (konstr-blatt-menge (rest sh-paare)))))
```