5. Übungsblatt zu Programmiersprachenkonzepte

Manuel Schwarz, Michael Stypa

12. Januar 2011

Aufgabe 5.1

```
(defun myreverse (l1)
    (cond ((null l1) nil)
          ((listp (first l1)) (append (myreverse (rest l1))
                                      (list(myreverse (first l1)))))
          (T (append (myreverse (rest l1)) (list (first l1))))
    )
)
```

Aufgabe 5.2

1. (a) `(format T "Das Wort heisst ~S." "LISP")`

   ```
   Das Wort heisst "LISP".
   NIL
   ```

   (b) `(format T "Das Symbol ist ~&~A." "LISP")`

   ```
   Das Wort heisst
   LISP.
   NIL
   ```

   (c) `(format T "Die Formatierung ~~S erzeugt diese ~A." 'Ausgabe)`

   ```
   Die Formatierung ~S erzeugt diese AUSGABE.
   NIL
   ```

   (d) `(format T "Die Formatierung ~~~S sieht anders aus." A)`
   Bei uninitialisiertem A: Fehler
   Sonst:

   ```
   Die Formatierung ~INHALT sieht anders aus.
   NIL
   ```

2. `(format t "Laenge: ~9,2f~%Breite: ~9,2f ~%Hoehe: ~9,2f~%" "22.34" "134.20" "1.0")`

(format t "Laenge: 9,2f

Aufgabe 5.3

```
(defun liesDatei (datei)
  (let ((stream (open datei))
        (liste (list '0)))
        (loop (setf zahl (read stream NIL))
              (if (eq zahl NIL)
                  (return (if (close stream) liste))
              )
              (push zahl liste)
        )
  )
)

(defun summe (liste)
  (cond ((equal (cdr liste) NIL) (car liste))
        (T (+ (car liste) (summe (cdr liste))))
  )
)

(defun anz (liste)
  (cond ((equal liste '()) 0)
        ((equal (cdr liste) NIL) 1)
        (T (+ 1 (anz (cdr liste))))
  )
)

(defun durchschnitt (datei)
  (setf liste (liesDatei datei))
  (setf divisor (anz liste))
  (if (equal divisor 0) 0 (/ (summe liste) divisor))
)
```

Aufgabe 5.4

```
(defun xxx (list)
  (do ((lst (cdr list)(cdr lst))
       (head (car list)(car lst))
       (lasthead list head)
      )
      ((null lst) NIL)
      (if (eq head lasthead)
          (return lst)
      )
  )
)
```

Die Funktion `xxx` bekommt eine Liste übergeben und prüft, ob zweimal hintereinander das gleiche Element aufgeführt wird. Ist dies der Fall, so wird alles (der Rest) nach dem ersten doppelten Element zurückgegeben. Gibt es so eine Dopplung nicht, so wird NIL zurückgegeben.

Beispiel:
`(xxx '(1 2 3 4 5)) = NIL`
`(xxx '(1 2 3 3 4 5)) = (4 5)`

Funktionalität der `do` -Funktion (Iteration):
Im ersten Teil wird festgelegt, welche Variablen es gibt, wie ihr Initialwert lautet und wie sie zu aktualisieren sind.
Im zweiten Teil wird eine Abbruchbedingung sowie ein return-Wert festgelegt und der dritte Teil ist der body der Funktion (der in diesem Fall ebenfalls einen return-Wert enthält, welcher jedoch bedingt ist).