Hochschule München
Fakultät für Informatik und Mathematik
M. Ruckert

Blatt 3

Theoretische Informatik II
Übungen

Aufgabe 11. Beschreiben Sie den Ablauf des folgenden Programs und erklären sie seine Ausgabe.

```
(define (apply-to f x) (f x))

(apply-to (lambda (x)
             (display "world !")
             (display x)
             (newline))
          (display "hello "))
```

Aufgabe 12. Schreibe eine Funktion `double`, die eine Funktion f als Argument nimmt und eine Funktion zurückgibt, die gleich der doppelten Anwendung von f ist. D.h für jede Funktion f und jede Zahl n sollte `((double` f `)` n `)` dasselbe Ergebnis liefern wie `(` f `(` f n `))`

Aufgabe 13. Schreibe eine Funktion `compose`, die zwei Funktionen f und g als Argument nimmt und eine Funktion zurückgibt, die die Hintereinanderausführung von f und g darstellt. D.h für jede Funktion f, jede Funktion g und jede Zahl n sollte `((compose` f g`)` n`)` dasselbe Ergebnis liefern wie `(` f `(` g n `))`

Aufgabe 14. Ein Polynom $a_nX^n + \cdots + a_1X + a_0$ mit $a_n \neq 0$ sei als Liste $(a_0\ \ldots\ a_n)$ dargestellt (man beachte die Reihenfolge!). Das Nullpolynom wird durch die leere Liste () repräsentiert.

Schreiben Sie eine Funktion `(polynom-funktion p)` die zu einem Polynom p, dargestellt durch die Liste `p`, die entsprechende einstellige Scheme Funktion, die p berechnet, als Wert liefert.

Beispiel:

```
(define p1 (polynom-funktion '(1 1)))    ; p1(x) = x + 1
(define p2 (polynom-funktion '(1 0 1)))    ; p2(x) = x*x + 1

(p1 2) ; liefert 2+1 = 3
(p2 2) ; liefert 2*2+1 = 4+1 = 5
```

Hinweis:

- Einfache rekursive Erzeugung der Polynomfunktion:

  Ist $p = a_nX^n + \cdots + a_1X + a_0$ mit $a_n \neq 0$ also `p` $=$ `(` $a_0, a_1\ \ldots\ a_n$ `)` so ist $p = qX + a_0$ mit $q = a_nX^{n-1} + \cdots + a_1$ und q wird dargestellt durch `q` $=$ `(` $a_1\ \ldots\ a_n$ `)`. Mittels `lambda` läßt sich nun die gewünschte Funktion einfach rekursiv erzeugen.

- Rekursive Konstruktion des Polynomterms:

  Instruktiv ist auch das folgende Vorgehen: Man erzeuge analog zur vorigen Löesung eine Scheme Struktur, die den Polynom-Term darstellt, also `(+` a_0 `(* X ( ... )))`. Hat man die Funktion `polynom-term` so erzeuge man die Liste `(lambda (X)` *Polynomterm* `)` und verwende die Funktion `eval`, die einen Scheme Ausdruck (hier die Liste) in den entsprechenden Wert (hier die Funktion) umwandelt.