(FP-Aufgaben 04)

Aufgabe 1: Welchen Typ hat die folgende Funktion f?

```
f  0  x  y = y ++ [x]
f  x  y  z = x : y : z
```

Aufgabe 2: Was ist das Ergebnis von (f1 0 3) und (f1 2 3)?

```
f1  0  x = let f2  x  y = x + y  in  f2  x  1
f1  x  y = x * f2  y
        where
            f2  x = x + y
```

Aufgabe 3: Schreiben Sie eine Funktion, die einen Satz als Parameter nimmt, und alle e's durch X ersetzt.

a) Definieren Sie diese Funktion mit einem **linear-rekursiven Prozess**.
b) Definieren Sie die Funktion mit einem **endständig-rekursiven** Prozess.

z.B. Der Aufruf
ersetzen „Die Ebene"
muss zurückliefern
„DiX EbXnX"

Aufgabe 4: Für die folgende Funktion:

$$fib(n) = \begin{cases} 0 & falls\ n = 0 \\ 1 & falls\ n = 1 \\ fib(n-1) + fib(n-2) & falls\ n \in \mathrm{N}\ und\ n > 1 \end{cases}$$

a) Definieren Sie eine Funktion (fib_rek n), die die Funktion fib in einem **baum-rekursiven Prozess** berechnet.

b) Definieren Sie eine Funktion (fib_iter n), die die Funktion fib in einem **iterativen Prozess** und mit Hilfe der Akkumulator-Technik berechnet.

Hinweis: Benutzen Sie zwei Akkumulatoren, die mit ersten zwei Fibonacci-Zahlen (bzw. 0 und 1) initialisiert sind.

c) Testen Sie Ihr Programm und vergleichen Sie die Laufzeit der Funktionen.

Hinweis: Sie können die Laufzeit einer Funktion mit dem folgenden Befehl in einem Terminal anzeigen lassen:
:set +s

Anhang:

Um einen Fehler zu lokalisieren, möchte man manchmal Zwischenergebnisse in einem Terminal ausgeben. Dafür kann man die trace-Funktion von Haskell benutzen. Diese Funktion hat den folgenden Typ:

```
trace :: String -> a -> a
```

Als Beispiel kann man im Rumpf der Funktion f mit den folgenden Gleichungen

```
f  0 = 1
f  n = n * f (n - 1)
```

die Werte, die an n gebunden werden, anzeigen lassen:

```
module  Fakultaet  where
import  Debug.Trace
f  0  =  1
f  n  =  trace ("n = " ++ show  n) (n) * f  (n - 1)
```

Die Funktion show mit dem folgenden Typ

```
show :: Show a => a -> String
```

kann jeden Wert in String umwandeln.