## Lernziele

- Higher order functions

### Aufgabe 1

Passen Sie die logger Funktion aus den Folien so an, dass sie auch rekursive Aufrufe der übergebenen Funktion (als Funktional) "loggt".

### Aufgabe 2

Gegeben sind folgende Typen:

```
// Binary tree
type 'a bTree = E | Nb of 'a bTree * 'a * 'a bTree

// List tree
type 'a lsTree = Nls of 'a * 'a lsTree list
```

Implementieren sie für beide Baumtypen jeweils eine `fold` Funktion. Bedenken sie dabei, dass die Idee eines foldes darin besteht, jeweils die Konstruktoren des betreffenden Typs durch mitgegebene Funktionen zu ersetzen. Zum Vergleich nochmals eine Implementation von einem `fold` für Listen:

```
let rec fold consReplacement empty = function
   | [] // replace empty list with empty-value
      -> empty
   | x::xs // replace Cons(x,xs) with consReplacement
      -> consReplacement x (fold consReplacement empty xs)
```

Verwenden Sie schliesslich Ihre fold-Funktionen um für die gegebenen Baumtypen je eine `map-` und eine `depth`-Funktion[1] zu implementieren.

[1] `map` soll dabei die Elemente des Baumes entsprechend einer gegebenen Funktion ersetzen und `depth` soll die Tiefe eines gegebenen Baumes berechnen.