Lernziele

- Der Funktionstyp
- Currying
- Options

Aufgabe 1

Bestimmen Sie (ohne FSI) die Typen von `x1,x2,x3`:

```
let x1 y z = if y z then 1 else 0
let x2 y z = if z y then 1 else 0
let x3 y z = List.filter y (z y)
```

Aufgabe 2

Erklären Sie in Ihren eigenen Worten den Unterschied der folgenden Definitionen:

$$f_1(x, y) = 2 \cdot x + 3 \cdot y$$
$$f_2(x)(y) = 2 \cdot x + 3 \cdot y.$$

Geben Sie Wertebereich und Definitionsbereich der Funktionen f_1 und f_2 an. Diskutieren Sie weiter die Aussage:

$$f_2(3) = f_1(3, \_).$$

Aufgabe 3

Erklären Sie wieso man in der funktionalen Programmierung "Curryied" Funktionen vorzieht um "generischen" Code zu schreiben.

Aufgabe 4

Herr "Norbert Thomas Forli" hat eine furchtbar langweilige und sinnlose Arbeit; er muss viele hundert mal pro Tag eine Funktion (vgl. Unten) berechnen und das Resultat in eine Tabelle eintragen. Um sich das Leben zumindest etwas leichter zu machen, hat er sich das folgende Konsolenprogramm geschrieben:

```
let rec calculate () =

   printfn "x = "
   let x = System.Console.ReadLine()

   printfn "y = "
   let y = System.Console.ReadLine()

   let result =
      2*(int x)/if (int y)%17 = 0 then 0 else (int y)
      |> float
      |> System.Math.Cos

   printfn "result = %f" result

   printfn "more? [y/n]"

   match System.Console.ReadLine().ToLower() with
   | "y" -> calculate ()
   | _ -> printfn "au revoir!"


calculate ()
```

Leider "stürzt" sein Programm oft ab, was zur Folge hat das er es neu starten muss und damit Zeit verliert. Benutzen Sie Option-Typen um das Programm von Herr Forli so zu modifizieren, dass es bei einer falschen Eingabe oder einer Division durch Null keine Exception wirft, sonder als Resultat "no valid result" zurückgibt und weiterläuft.

Aufgabe 5

Implementieren Sie ein simples "Black-Jack" Konsolenprogramm wie in der Vorlesung besprochen.