Summary functions:

- Functions are data! (table lookup)
- Functions returning functions are just multivariate functions:
  - $\lambda x.\, \lambda y. = \lambda\, x\, y.$
  - fun x -> fun y -> ... = fun x y -> ...
  - f : a -> b -> c
- Multivariate functions can be partially applied:

```
let f x y = x**2 + 3*y + 5
let g = f 5   => g 4 = 42

map (f 3) [1; 2; 3]
(map : (a -> b) -> [a] -> [b])
```

- Higher-order functions are functions taking functions as arguments:

  $\lambda f.\, \lambda x.\, f\ x$

  ex. let f g x = g (2 * x + 1)

  map g y

- HOF:s are extremely important in FP, and are one of the main devices for abstraction!

- Multivariate functions are equivalent to functions of a tuple:

  $f : a \to b \to c \cong f' : (a, b) \to c$

  let curry (f : (a, b) → c) x y = f (x, y)

  let uncurry (f : a → b → c) (x, y) = f x y

- Warning: partial functions!
  - List.head, tail, find
  - sqrt
  - ...

Types in F#

- Simple terms:

  a, b, c : generic, not yet inferred
  'a, 'b, 'c : explicitly generic

  int, float, string...

- Algebraic: int option, string list
  array<float>

- Functions:

  f : a → b

  f' : int → int

List.map : (a → b) → a list → b list

(>>) : (a → b) → (b → c) → a → c

(|>) : a → (a → b) → b

let (>>) f g x = g (f x) [g ∘ f (x)]
= x |> f |> g

let (<<) f g x = f (g x) [f ∘ g (x)]

INF-3910 Lecture 3 (4)

- λ-calculus is a formal system in logic, based solely on the abstraction and application of λ-terms.

- λ-calculus is an universal model of computation, equivalent to any Turing machine.

- Two main variants: untyped and singly typed λ-calculus.

- untyped λ-calculus is stronger than typed, but can lead to paradoxes. Less can be proved about untyped λ-calc, e.g. termination (halting problem)

- Using λ-calculus we can encode anything:
  - Nat -> Q -> R -> C
  - and, or, not
  - bool
  - pairs, lists ...

- Pure λ-calculus is tedious as f*ck!

- Curch encoding natural numbers:

$\lambda f x . x := 0$
$\lambda f x . f x := 1$
$\lambda f x . f (f x) := 2$
$\vdots$

- Bool: $T := \lambda x y . x$
$F := \lambda x y . y \quad (= 0)$

- Pairs: $pair := \lambda x y f . f x y$
$fst := \lambda p . p T$
$snd := \lambda p . p F$
$nil := \lambda x . T$

$rep := \lambda n x . n\ pair\ x\ nil$
$succ := \lambda n f x . f (n f x)$
$plus := \lambda m n . \lambda f x . m f (n f x)$
$\Rightarrow \lambda m n . m\ succ\ n$
$ifthenelse := \lambda p . \lambda a . \lambda b . p\ a\ b$
(p: bool)