made light of

Typen in OCaml

`int`	vorzeichenbehafteter Integer
`float`	vorzeichenbehafteter Gleitkommazahl
`bool`	Boole'scher Wert: `true` oder `false`
`char`	Schriftzeichen
`string`	Zeichenkette
`'a list`	Liste, deren Elemente Typ 'a haben (`int list` = Integer Liste)
`unit`	nichts; der Typ des Resultats Funktionen, die kein Objekt zurück geben (leer).

```
type expression =
Var of string
| Const of int
| Add of expression * expression
| Mul of expression * expression
```

Selbstdefinierter (hier rekursiver) Typ (Beispiel).

z.B.: `# Add ( Mul (Const 2 , Var "x") , Var "y");;` $\rightsquigarrow$ $2x + y$

OCaml Toolkit

`#use "filename";;`	Die Inhalte der Datei "filename" evaluieren.
`let x=4 and y=5;;`	Die Variablen `x` und `y` global definieren.
`let x=9 in 3 * x;;`	Die Variable `x` lokal in der expression `3 * x` definieren.
`+, - , *, /`	Integer Addition, Substraktion, Multiplikation, Division
`+., -., *., /.`	Float Addition, Substraktion, Multiplikation, Division
`<, >, <=, >=`	Vergleichen von Zahlen, Zeichen, alles was geordnet ist.
`"OCaml"`	Die Zeichenkette `OCaml`.
`'a'`	Das Zeichen `a`
`String.length "Logic"`	Länge der Zeichenkette "Logic" `( - :  int = 5)`
`"pro" ^ "of" = "proof"`	Zeichenkettenkonkatenation
`"\n"`	Zeilenende bei Zeichenkettem.
`||`	Boole'scher „oder"
`&&`	Boole'scher „und"
`[]`	Die leere Liste (auch `Nil`)
`3::[4; 5; 7] = [3; 4; 5; 7]`	Am Anfang einer Liste einfügen.
`3::4::5::7::[] = [3; 4; 5; 7]`	Äquivalente Darstellungen einer Liste.
`List.length [1; 2; 3] = 3`	Länge der Liste [1; 2; 3]
`fun x -> x + 1;;`	Die anonyme Funktion $x \mapsto x+1$.
`let f x = x + 1;;`	Eine Funktion definieren.
`let f = fun x -> x + 1;;`	Die gleiche Funktion wie oben definieren.

`let rec factorial n =`
`  if n<=0 then 1`
`  else n * factorial (n-1);;`
	Eine rekursive Funktion definieren.

`f 2;;`	Funktionsaufruf der Funktion `f` mit dem Wert `2`.
`f ();;`	Funktionsaufruf def Funktion `f` ohne Parametern.

`let rec last list =`
`  match list with`
`  [] -> []`
`  | [a] -> [a]`
`  | first::rest -> last (rest);;`
	Rekursives „pattern matching" auf eine Liste