OCaml
made light of

„Programs should be written for humans to read, and only incidentally for machines to execute.“
– Hal Abelson & Jay Sussman

Bevor Sie mit den Aufgaben anfangen, schauen Sie sich bitte die OCaml-Richtlinien an:
`https://ocaml.org/learn/tutorials/guidelines.html`

Aufgabe 1.
Schreiben Sie für jede der folgenden Funktionen eine OCaml-Aussage, die das Ergebnis `- : int = 9` liefert:

```
let f1 = fun x -> x * x
and f2 = fun x y -> x / y
and f3 = fun x -> (fun y -> x / y)
and f4 x = x 3
and f5 x = if x=2
           then (fun () -> x)
           else (fun () -> x * 3);;
```

Aufgabe 2.
Schreiben Sie für jede der folgenden Beispiele mindestens eine, nicht-triviale Funktion, die das gleiche Ergebnis liefert:

a) `append_to_end_int_list 0 [1; 2; 3; 4] = [1; 2; 3; 4; 0]`

b) `append_to_end_string 's' "hacker" = "hackers"`

c) `order "theorem prover" = "eeehmooprrrtv"` (Leerzeichen entfernen)

d) `reverse_int [1; 2; 3] = [3; 2; 1]`

e) `reverse_string "abc" = "cba"`

f) `palindrome_p "never odd or even" = true` (Leerzeichen ignorieren)

Sie können die Ressources unten, das Internet, und folgene Funktionen dafür benutzen:

```
let explode string =
  let rec expl a b =
    if a < 0 then b
    else expl (a - 1) (string.[a] ::  b)
  in
  expl (String.length string - 1) [];;
```

```
let implode list =
  let result = String.create (List.length list)
  in
  let rec impl n = function
    | [] -> result
    | a ::  list -> result.[n] <- a;
                    impl (n + 1) list
  in
  impl 0 list;;
```

Ressourcen z.B.: `http://caml.inria.fr/pub/docs/manual-ocaml/`

(weiter auf der nächsten Seite)

Aufgabe 3. „finite partial functions"
Experimentieren Sie mit den folgenden Funktionen in die REPL, und versuchen Sie herauszufinden, was sie machen oder wie sie funktionieren.

```
# graph (3 |=> 4);;
# graph (3 |-> 4);;
# let smallsqs = fpf [1;2;3] [1;4;9];;
# defined smallsqs 3;;
# graph (undefine smallsqs 3);;
# tryapply smallsqs 6 500;;
# tryapplyl smallsqs 15;;
# dom smallsqs;;
# ran smallsqs;;
# let empt = undefined;;
# graph undefined;;
```

Aufgabe 4.
Schauen Sie den Code des ersten Kapitels an: `http://www.math.uni-bonn.de/ag/logik/teaching/2015WS/hauptseminar-dateien/Harrison-Kapitel1/` und versuchen Sie die entsprechenden Aufgaben auf Seite 24.