```
# let inc x = x + 1 ;;
val inc : int -> int = <fun>
# let add x y = x + y ;;
val add : int -> int -> int = <fun>
# let two x = 2 ;;
val two : 'a -> int = <fun>
# let id x = x ;;
val id : 'a -> 'a = <fun>
# let ti x = (x,2) ;;
val ti : 'a -> 'a * int = <fun>

# let pair a b = (a,b) ;;
val pair : 'a -> 'b -> 'a * 'b = <fun>
# let fst (a,b) = a ;;
val fst : 'a * 'b -> 'a = <fun>
# let snd (a,b) = b ;;
val snd : 'a * 'b -> 'b = <fun>

# let app f x = f x ;;
val app : ('a -> 'b) -> 'a -> 'b = <fun>
# let ppa x f = f x ;;
val ppa : 'a -> ('a -> 'b) -> 'b = <fun>
# let compose f g x = f (g x) ;;
val compose : ('a -> 'b) -> ('c -> 'a) -> 'c -> 'b = <fun>

# let pair2 a b f = f a b;;
val pair2 : 'a -> 'b -> ('a -> 'b -> 'c) -> 'c = <fun>

# let prawda x y = x ;; (* fst2 *)
val prawda : 'a -> 'b -> 'a = <fun>
# let falsz x y = y ;; (* snd2 *)
val falsz : 'a -> 'b -> 'b = <fun>

# let nie f a b = f b a ;; (* flip *)
val nie : ('a -> 'b -> 'c) -> 'b -> 'a -> 'c = <fun>
# let i f g a b = f (g a b) b ;;
val i : ('a -> 'b -> 'c) -> ('d -> 'b -> 'a) -> 'd -> 'b -> 'c = <fun>
# let lub f g a b = f a (g a b) ;;
val lub : ('a -> 'b -> 'c) -> ('a -> 'd -> 'b) -> 'a -> 'd -> 'c = <fun>
# let xor f g a b = f (g b a) (g a b) ;;
val xor : ('a -> 'a -> 'b) -> ('c -> 'c -> 'a) -> 'c -> 'c -> 'b = <fun>

# let s x y z = x z (y z) ;;
val s : ('a -> 'b -> 'c) -> ('a -> 'b) -> 'a -> 'c = <fun>
# let callCC f k = f (fun c d -> k c) k ;;
val callCC : (('a -> 'b -> 'c) -> ('a -> 'c) -> 'd) -> ('a -> 'c) -> 'd = <fun>
```