

Objective Caml version 3.06

```

# let rec member x l =
  match l with
  | [] -> false
  | x :: l' -> true
  | _ :: l' -> member x l';;
  Characters 84-91:
Warning: this match case is unused.
  | _ :: l' -> member x l';;
  ^^^^^^^^^
val member : 'a -> 'b list -> bool = <fun>

# member 3 [1;2;3;4];;
- : bool = true

# member 5 [1;2;3;4];;
- : bool = true

# let rec member x l =
  match l with
  | [] -> false
  | y :: l -> x = y || member x l;;
  val member : 'a -> 'a list -> bool = <fun>

# member 3 [1;2;3;4];;
- : bool = true

# member 5 [1;2;3;4];;
- : bool = false

# List.mem;;
- : 'a -> 'a list -> bool = <fun>

# List.mem 3 [1;2;3;4];;
- : bool = true

# List.mem 5 [1;2;3;4];;
- : bool = false

# let rec append l1 l2 =
  match l1 with
  | [] -> l2
  | x :: l -> x :: append l l2;;
  val append : 'a list -> 'a list -> 'a list = <fun>

# append [1;2;3;4] [1;2;3;4];;
- : int list = [1; 2; 3; 4; 1; 2; 3; 4]

# List.append;;
- : 'a list -> 'a list -> 'a list = <fun>

# (@);;
- : 'a list -> 'a list -> 'a list = <fun>

# [1;2;3;4] @ [1;2;3;4];;
- : int list = [1; 2; 3; 4; 1; 2; 3; 4]

# let rec reverse l =
  match l with
  | [] -> []
  | x :: l' -> reverse l' @ [x];;
  val reverse : 'a list -> 'a list = <fun>

# reverse [1;2;3;4];;
- : int list = [4; 3; 2; 1]

# let reverse l =
  let rec reverse_append l1 l2 =
    match l1 with
    | [] -> l2
    | x :: l' -> reverse_append l' (x :: l2)
  in reverse_append l [];;
  val reverse : 'a list -> 'a list = <fun>

# reverse [1;2;3;4];;
- : int list = [4; 3; 2; 1]

```

```
# List.rev;;
- : 'a list -> 'a list = <fun>

# List.rev_append;;
- : 'a list -> 'a list -> 'a list = <fun>

# let rec map f l =
  match l with
  [] -> []
  | x :: l' -> f x :: map f l';;
  val map : ('a -> 'b) -> 'a list -> 'b list = <fun>

# map (fun x -> x * x) [1;2;3;4];;
- : int list = [1; 4; 9; 16]

# let rec exists p l =
  match l with
  [] -> false
  | x :: l' -> p x || exists p l';;
  val exists : ('a -> bool) -> 'a list -> bool = <fun>

# let member x l = exists (fun y -> y = x) l;;
val member : 'a -> 'a list -> bool = <fun>

# member 3 [1;2;3;4];;
- : bool = true

# member 5 [1;2;3;4];;
- : bool = false

# let member x = exists (fun y -> y = x);;
val member : 'a -> 'a list -> bool = <fun>

# member 3 [1;2;3;4];;
- : bool = true

# member 5 [1;2;3;4];;
- : bool = false

# let rec for_all p l =
  match l with
  [] -> true
  | x :: l' -> p x && for_all p l';;
  val for_all : ('a -> bool) -> 'a list -> bool = <fun>

# let constant l =
  match l with
  [] -> true
  | x :: l' -> for_all (fun y -> y = x) l';;
  val constant : 'a list -> bool = <fun>

# constant [1;2;3;4];;
- : bool = false

# constant (map (fun x -> 1) [1;2;3;4]);;
- : bool = true

# let rec filter p l =
  match l with
  [] -> []
  | x :: l' -> if p x then x :: filter p l' else filter p l';;
  val filter : ('a -> bool) -> 'a list -> 'a list = <fun>

# filter (fun x -> x mod 2 = 0) [1;2;3;4];;
- : int list = [2; 4]

#
```