

Objective Caml version 3.06

```
# type date = int * int * int;;
type date = int * int * int

# type weekday = string;;
type weekday = string

# type day = weekday * date;;
type day = weekday * date

# let today: day = ("Donnerstag", 29, 4, 2004);;
Characters 18-43:
  let today: day = ("Donnerstag", 29, 4, 2004);;
                    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
This expression has type string * int * int * int but is here used with type
  day = weekday * date

# let today: day = ("Donnerstag", (29, 4, 2004));;
val today : day = ("Donnerstag", (29, 4, 2004))

# let holiday ((w, d): day) = w = "Sonntag";;
val holiday : day -> bool = <fun>

# holiday today;;
- : bool = false

# let holiday ((w, _): day) = w = "Sonntag";;
val holiday : day -> bool = <fun>

# holiday today;;
- : bool = false

# let holiday = fun ((w, _): day) -> w = "Sonntag";;
val holiday : day -> bool = <fun>

# holiday today;;
- : bool = false

# let holiday = fun ("Sonntag", _) : day -> true;;
Characters 14-47:
Warning: this pattern-matching is not exhaustive.
Here is an example of a value that is not matched:
("", _)
  let holiday = fun ("Sonntag", _) : day -> true;;
                    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
val holiday : day -> bool = <fun>

# holiday today;;
Exception: Match_failure ("", 14, 47).

# let holiday = fun ("Sonntag", _) : day -> true || _ -> false;;
Characters 48-49:
  let holiday = fun ("Sonntag", _) : day -> true | _ -> false;;
                    ^
Syntax error

# let holiday = function ("Sonntag", _) : day -> true | _ -> false;;
val holiday : day -> bool = <fun>

# holiday today;;
- : bool = false

# let holiday (d: day) = match d with ("Sonntag", _) -> true | _ -> false;;
val holiday : day -> bool = <fun>

# holiday today;;
- : bool = false

# let holiday (d: day) = let (w, _) = d in w = "Sonntag";;
val holiday : day -> bool = <fun>

# holiday today;;
- : bool = false
```

```

# let holiday (d: day) =
  try let ("Sonntag", _) = d in true with Match_failure _ -> false;;
  Characters 34-46:
Warning: this pattern-matching is not exhaustive.
Here is an example of a value that is not matched:
("", _)
  try let ("Sonntag", _) = d in true with Match_failure _ -> false;;
  ^^^^^^^^^^^^^^^^^
val holiday : day -> bool = <fun>

# holiday today;;
- : bool = false

# let holiday (d: day) = fst d = "Sonntag";;
val holiday : day -> bool = <fun>

# holiday today;;
- : bool = false

# let tomorrow: day = ("Freitag", (30, 4, 2004));;
val tomorrow : day = ("Freitag", (30, 4, 2004))

# today < tomorrow;;
- : bool = true

# let before (d1: day) (d2: day) =
  let (_, (d1, m1, y1)) = d1
  and (_, (d2, m2, y2)) = d2
  in y1 < y2 || (y1 = y2 && (m1 < m2 || (m1 = m2 && d1 < d2)));;
  val before : day -> day -> bool = <fun>

# before today tomorrow;;
- : bool = true

# before tomorrow today;;
- : bool = false

# before today today;;
- : bool = false

# let before ((_, (d1, m1, y1)): day) ((_, (d2, m2, y2)): day) =
  y1 < y2 || (y1 = y2 && (m1 < m2 || (m1 = m2 && d1 < d2)));;
  val before : day -> day -> bool = <fun>

# before today tomorrow;;
- : bool = true

# before tomorrow today;;
- : bool = false

#

```