```
        Objective Caml version 3.06

# type date = {day: int; month: int; year: int};;
type date = { day : int; month : int; year : int; }

# type weekday = So | Mo | Di | Mi | Do | Fr | Sa;;
type weekday = So | Mo | Di | Mi | Do | Fr | Sa

# So;;
- : weekday = So

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

# let today: day = (Do, {day = 29; month = 4; year = 2004});;
val today : day = (Do, {day = 29; month = 4; year = 2004})

# let now: day = (Do, {year = 2004; month = 4; day = 29});;
val now : day = (Do, {day = 29; month = 4; year = 2004})

# now = today;;
- : bool = true

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

# let holiday = fun ((So, _): day) -> true;;
Characters 14-40:
Warning: this pattern-matching is not exhaustive.
Here is an example of a value that is not matched:
((Sa|Fr|Do|Mi|Di|Mo), _)
  let holiday = fun ((So, _): day) -> true;;
                ^^^^^^^^^^^^^^^^^^^^^^^^^
val holiday : day -> bool = <fun>

# let workday = fun (((Mo | Di | Mi | Do | Fr), _): day) -> true;;
Characters 14-62:
Warning: this pattern-matching is not exhaustive.
Here is an example of a value that is not matched:
((Sa|So), _)
  let workday = fun (((Mo | Di | Mi | Do | Fr), _): day) -> true;;
                ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
val workday : day -> bool = <fun>

# holiday today;;
Exception: Match_failure ("", 14, 40).

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

# let tomorrow: day = (Fr, {day = 30; month = 4; year = 2004});;
val tomorrow : day = (Fr, {day = 30; month = 4; year = 2004})

# before today tomorrow;;
- : bool = true

# let before ((_, {day = day; month = month; year = year}): day)
             ((_, {day = day'; month = month'; year = year'}): day) =
  year < year' || (year' = year && (month < month' || (month = month' && day < day')));;
    val before : day -> day -> bool = <fun>

# before today tomorrow;;
- : bool = true

# let year (_, date) = date.year;;
val year : 'a * date -> int = <fun>

# year today;;
- : int = 2004

# let year d = (snd d).year;;
val year : 'a * date -> int = <fun>

# year today;;
- : int = 2004
```