1 Problem

You are given the following information, but you may prefer to do some research for yourself.

1. 1 Jan 1900 was a Monday.
2. Thirty days has September, April, June and November. All the rest have thirty-one, Saving February alone, Which has twenty-eight, rain or shine. And on leap years, twenty-nine.
3. A leap year occurs on any year evenly divisible by 4, but not on a century unless it is divisible by 400.

How many Sundays fell on the first of the month during the twentieth century (1 Jan 1901 to 31 Dec 2000)?

2 Solution

```
import qualified Data.Map as Map
import Data.Maybe
import Data.List
data DOW = Monday | Tuesday | Wednesday | Thursday | Friday | Saturday | Sunday
  deriving (Eq, Enum, Show, Ord)
data Month = January | February | March | April | May | June | July | August
    | September | October | November | December
  deriving (Eq, Enum, Show, Ord)
data DateSet = DateSet
   {day :: DOW
   , mon :: Month
   , dat :: Int
   , yr :: Int
   , sinceStart :: Int
   } deriving (Eq, Show)
days = cycle [Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday]
months = cycle [January, February, March, April, May, June, July, August
, September, October, November, December]
monthDayMap = Map.fromList
   [(January, 31)
   , (February, 28)
   , (March, 31)
   , (April, 30)
   , (May, 31)
   , (June, 30)
```

```
    , (July, 31)
    , (August, 31)
    , (September, 30)
    , (October, 31)
    , (November, 30)
    , (December, 31)
    ]
daysInMonth m y
    | m ≢ February   = fromJust $ Map.lookup m monthDayMap
    | y ‘mod‘ 400 ≡ 0 = 28
    | y ‘mod‘ 100 ≡ 0 = 29
    | otherwise        = 28
  -- nextMonth :: DateSet -¿ DateSet
nextMonth ds = DateSet dow′ mon′ 1 yr′ ss′
  where
    dayDiff = daysInMonth (mon ds) (yr ds)
    ss′ = (sinceStart ds) + dayDiff
    mon′ = nextMonth′ (mon ds)
    yr′ = if (mon ds) ≡ December
      then (yr ds) + 1
      else (yr ds)
    dow′ = days !! ss′
countSundayFirsts :: Int → DateSet → Int
countSundayFirsts ct ds
    | yr ds > 2000       = ct
    | day ds ≡ Sunday = countSundayFirsts (1 + ct) (nextMonth ds)
    | otherwise           = countSundayFirsts ct (nextMonth ds)
nextMonth′ mon = months !! k
  where ky = Map.keys monthDayMap
        k′ = fromJust $ elemIndex mon ky
        k = (k′ + 1) ‘mod‘ 12
main = do
  let startDay = DateSet Tuesday January 1 1901 1
      mycount = countSundayFirsts 0 startDay
  putStrLn $ "During the twentieth century, " ++ show mycount ++ " Sundays fell on the firs
```

3 Result

```
runhaskell problem19.lhs
During the twentieth century, 171 Sundays fell on the first of the month.
```