課題 6

カレンダーを作る関数 year と month を定義しなさい。

解答及び解説

はじめに、1ヶ月分のカレンダーを作る。後々のために、タイトルとボディに分けておく。タイトルは、「何年何月」という部分で、ボディは、「曜日と日付」が並ぶ。

ボディは、複数行で構成されるので、行ごとに文字列として、それをリストにする。1 行目は、いつも同じ曜日なので、mHead という定数とする。ボディの残りの部分は、まず whatYoubi でその月の「ついたち」が何曜日であるかを求める。但し、日曜を 0 とする数値で曜日を表すようにする。それを使って、ついたちが始まるまでの空白部分を埋めるために fspace を作る。各日はすべて 3 文字 (隣との隙間含む) 分を占めるようにするため、曜日の 3 倍分のスペースになる。それに続いて、その月の日数分の日付 mdays を作成する。最後に、最終週の最後 (土曜の欄) までを空白で埋めるための dpad を付け足す。こうして作られた文字列を先頭から、1 週分ずつに切り分け、必ず 6 週分となるようにする。(状況により最後の行は空白のみとなる)

```
month :: Month -> Year -> IO ()
month m y = display $ mtitle m y : mbody m y

mtitle :: Month -> Year -> String
mtitle m y = "      "++(mStr m)++" "++(yStr y)

mbody :: Month -> Year -> [String]
mbody m y = mHead:mbody' (whatYoubi 1 m y) (monthDays m y)

mbody' :: Int -> String -> [String]
mbody' n mdays = splitWeek 6 (fspace ++ mdays ++ dpad)
    where fspace = makeSpace (3*n) ""
```

こうして、1ヶ月のカレンダーができる。補助として使った関数は、以下の通りである。

```
monthDays :: Month -> Year -> String
monthDays m y = take (3*(mdays m y)) dayString
    where mdays m y = if m == 2 && leapYear y then 29
                      else [31,28,31,30,31,30,31,31,30,31,30,31]!!(m-1)

splitWeek :: Int -> String -> [String]
splitWeek 0 d = []
splitWeek n d = w:splitWeek (n-1) e
```

```
          where (w, e) = splitAt 21 d

makeSpace :: Int -> String -> String
makeSpace 0 s = s
makeSpace n s = makeSpace (n-1) (' ':s)

yStr :: Year -> String
yStr y = show y

mStr :: Month -> String
mStr m | m < 10    = (' ':(show m))++"月"
       | otherwise = (show m)++"月"
mHead :: String
mHead = " 日 月 火 水 木 金 土"

dayString :: String
dayString = "  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23

dpad :: String
dpad = "                                                    "
```

次に、1 年分のカレンダーを作る。年数をタイトルにして、12ヶ月分のボディが続く。(タイトルとボディの間には空行を挿入する。) ボディは、3ヶ月を単位として、4 つを並べる。

```
year :: Year -> IO ()
year y = display $ ytitle y : "" : ybody y

ytitle :: Year -> String
ytitle y = "                                "++(yStr y)

ybody :: Year -> [String]
ybody y = quoter 0 y

quoter :: Month -> Year -> [String]
quoter 4 y = []
quoter n y = (qhead n : makeQuoter n 1 y) ++ (quoter (n+1) y)

qhead :: Month -> String
qhead n = "        " ++ (mStr (3*n+1))
```

```
    ++ "                    " ++ (mStr (3*n+2))
    ++ "                    " ++ (mStr (3*n+3))


makeQuoter :: Month -> Int -> Year -> [String]
makeQuoter n 3 y = (mbody (3*n+3) y)
makeQuoter n m y = merge (mbody (3*n+m) y) (makeQuoter n (m+1) y)
    where merge [] [] = []
       merge (m:ms) (mm:mms) = (m++(' ':mm)) : merge ms mms
```

最後に、出来上がったカレンダーを表示する。month の定義を見ると、

```
month :: Month -> Year -> IO ()
month m y = display $ mtitle m y : mbody m y
```

となっており、$の後ろにある mtitle m y : mbody m y の部分では、単に文字列のリストを組み立てている。このことは、year の場合も同様である。

そして、関数 display によりリストにある順に、文字列を一行ずつ表示する。そのためのプログラムは、次の通りである。

```
display :: [String] -> IO ()
display []     = return ()
display (x:xs) = do putStrLn x
                    display xs
```

基本的な部分はここまでで終わり。残ったのは、日付に関する様々な型や関数のプログラムである。まずは、年月日を表す型の定義。これらは単純に整数とした。

```
type day = int
type month = int
type year = int
```

次に、ある年が閏年であるか否かを判定するプログラム。これには、日本におけて、グレゴリオ暦が採用された、明治 6 年 (1873 年) 以降についてのみ対応することとした。それ以前のカレンダーについては、曜日などが違っている可能性がある。

```
youbi1873Jan1 = 3


leapYear :: Int -> Bool
leapYear y | mod y 400 == 0 = True
           | mod y 100 == 0 = False
           | mod y   4 == 0 = True
           | otherwise      = False
```

```
whatYoubi :: Day -> Month -> Year -> Int
whatYoubi d m y = mod (dy + dm + d + youbi1873Jan1 - 1) 7
    where dm = month2days y (fromEnum m)
          dy = year2days y

year2days y = (y-1873)*365+(div(y-1873)4)-(div(y-1801)100)+(div(y-1601)400)

month2days y m | leapYear y  = if m > 2 then 1 + m2d m else m2d m
               | otherwise   = m2d m

m2d n = sum (take (n-1) mdays)
    where mdays = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30]
```

また、曜日については、前述したように日曜日を 0 で表し、以降月曜が 1、火曜が 2 と続き、土曜が 6 である。なお、明治 6 年 1 月 1 日は水曜日であるため、youbi1873Jan1 の値は、3 となっている。

さて、最後にリストの操作において、効率の悪い演算 (++) をできる限り排除するように一部の関数を修正する。

まずは、mbody' である。

```
mbody' :: Month -> Year -> [String]
mbody' n mdays = splitWeek 6 $ makeSpace (3*n) (mdays ++ dpad)
```

ここでは、mdays ++ dpad の部分でも演算 (++) を使っているが、mdays を求める関数 monthDays では、その月の日数によって日付の文字列を組み立てるのではなく、もともと最長 (31 日分) の日付を表す文字列 dayString から、必要な分のみ取り出す方法を使っている。そのため、dpad に必要な分の日付を追加して mdays を組み立てるよりは、今のまま (++) を使う方が良いと思う。

他には、関数 qhead に効率アップの余地がある。但し、これも分かりやすさでは、元のままの方が良いように思う。

```
qhead :: Month -> String
qhead n = makeSpace 8 $ mStrR (3*n+1) $ makeSpace 18 $
          mStrR (3*n+2) $ makeSpace 18 $ mStrR (3*n+3) ""

mStrR :: Month -> String -> String
mStrR n str = if n < 10 then (' ':show n)++('月':str)
                        else (show n)++('月':str)
```