SOFTWARE—PRACTICE AND EXPERIENCE, VOL. 23(4), 383–404 (APRIL 1993)

Calendrical Calculations, II: Three Historical Calendars

EDWARD M. REINGOLD AND NACHUM DERSHOWITZ
Department of Computer Science, University of Illinois at Urbana-Champaign, 1304 W. Springfield Avenue, Urbana, IL 61801–2987, U.S.A.

AND

STEWART M. CLAMEN
School of Computer Science, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213–3891, U.S.A.

SUMMARY

Algorithmic presentations are given for three calendars of historical interest, the Mayan, French Revolutionary, and Old Hindu.

KEY WORDS Calendar Mayan calendar French Revolutionary calendar Hindu calendar

From a chronological point of view the substitution for the mean calendaric system of one based on the true movements of the sun and moon, was anything but an improvement, as it destabilized the foundations of the time reckoning. Indeed, the system may have had the charm of adapting daily life as nearly as the astronomical knowledge permitted to the movement of the heavenly bodies, but on the other hand it broke ties with history, as there was no unity either of elements or systems. The very complexity of the system is a proof of its primitiveness.

—W. E. van Wijk (1938)

INTRODUCTION

In an earlier paper,[1] the first two authors gave algorithmic descriptions of five calendars in widespread use, namely the Gregorian (New Style), Julian (Old Style), Islamic (Moslem), Hebrew (Jewish), and ISO (International Organization for Standardization) calendars. (For an ancient comparison of the Islamic, Hebrew, and Julian calendars, see al-Biruni's classic treatise.[2]) In this paper, we treat similarly three calendars of historical interest: the Mayan, French Revolutionary (*le calendrier républicain*), and Old Hindu calendars. We follow the same conventions as in our

0038–0644/93/040383–22$16.00
Received 6 April 1992
Revised 26 October 1992

earlier paper, presenting all algorithms in the form of COMMON LISP[3] functions.* A brief explanation of Lisp, and why we chose it as a vehicle, was given in our earlier paper. As in that paper, we have chosen not to optimize the code at the expense of algorithmic clarity; consequently, improvements are possible in economy and in more appropriate use of Lisp functions.

Aside from their obvious historical value, the three calendars presented here are of interest in that they introduce algorithmic issues not present in the calendars of our previous paper. These new issues are applicable to a wide range of ancient and current calendars. There is also, certainly, an element of whimsy in our selection of ethnically, geographically, and chronologically diverse systems of chrononomy.

As described in our earlier paper, we have chosen Monday, January 1, 1 C.E.† (Gregorian) as our absolute day 1 and count forward day-by-day from there, taking the passage of time as a sequence of days numbered 1, 2, 3, ... that the various human-oriented calendars label differently. For example, absolute day 710,347 is called November 12, 1945 C.E. on the Gregorian calendar. All that is required for calendrical conversion is the ability to convert to and from this absolute calendar.

Our earlier paper gave Lisp functions to do the conversions for the Gregorian, ISO, Julian, Islamic, and Hebrew calendars. For example, to convert an absolute date to a Gregorian date we sequentially determine the year, month, and day of the month—the year is first closely approximated from below and then found precisely by stepping through subsequent years (that is, by a linear search).‡ The month is then found by a similar linear process, and the day of the month is determined by subtraction. The linear searches use a macro (from our previous paper) to compute sums. The expression

---

*To insure correctness, all code in this paper was typeset directly from working Lisp functions. We will gladly provide these Lisp functions in electronic form: send an empty electronic mail message to `reingold@cs.uiuc.edu` with the subject line containing precisely the phrase 'send-cal'; your message will be answered automatically with the combined code from this paper and our earlier one.

†Common era; or, A.D.

‡In our previous paper we noted that the exact determination of the Gregorian year from the absolute *date* can be viewed as an exercise in base conversion in a mixed-radix system, but (as pointed out to us by Jamshid Afshar), we gave an erroneous description of the conversion. Here is the correct description:

$d_0 \;= \mathit{date} - 1$	{prior days}
$n_{400} = d_0 \text{ div } 146097$	{number of completed 400 year cycles}
$d_1 \;= d_0 \text{ mod } 146097$	{prior days not included in n_{400}}
$n_{100} = d_1 \text{ div } 36524$	{number of 100 year cycles not included in n_{400}}
$d_2 \;= d_1 \text{ mod } 36524$	{prior days not included in n_{400} or n_{100}}
$n_4 \;= d_2 \text{ div } 1461$	{number of 4 year cycles not included in n_{400} or n_{100}}
$d_3 \;= d_2 \text{ mod } 1461$	{prior days not included in n_{400}, n_{100}, or n_4}
$n_1 \;= d_3 \text{ div } 365$	{number of years not included in n_{400}, n_{100}, or n_4}
$d_4 \;= d_3 \text{ mod } 365$	{prior days not included in n_{400}, n_{100}, n_4, or n_1}

if $(n_{100} = 4)$ or $(n_1 = 4)$ **then**
  date is ordinal day 366 in Gregorian year $400n_{400} + 100n_{100} + 4n_4 + n_1$
else
  date is ordinal day $d_4 + 1$ in Gregorian year $400n_{400} + 100n_{100} + 4n_4 + n_1 + 1$.

```
(sum f i k p)
```

computes $\sum_{i \geq k, p(i)} f(i)$; that is, the expression $f(i)$ is summed for all $i = k, k+1, \ldots$, continuing only as long as the condition $p(i)$ holds. The COMMON LISP definition of `sum` is

```
(defmacro sum (expression index initial condition)
;; Sum expression for index = initial and successive integers,
;; as long as condition holds.
  (let* ((temp (gensym)))
    `(do ((,temp 0 (+ ,temp ,expression))
          (,index ,initial (1+ ,index)))
         ((not ,condition) ,temp))))
```

In this paper we give functions for the Mayan, French Revolutionary, and Old Hindu calendars. Together with the functions in our previous paper, these functions enable conversion back and forth between any of the eight calendars. As in our earlier paper, the algorithms given in this paper do *not* generally work for non-positive absolute dates.

To simplify matters in our earlier paper, we altogether avoided dealing with dates before the common era. For example, our Lisp functions for the Hebrew calendar included the constant 1,373,429 as the number of days on the Hebrew calendar before our absolute date 0. This circumlocution avoided a statement such as 'the epoch of the Hebrew calendar is Saturday, September 5, -3760 (Gregorian)'. In the present paper, however, we deal with two calendars for which the scholarly literature is replete with such statements; thus, to aid the reader interested enough to pursue matters through further reading, we now explain how years before the common era are conventionally handled. This convention is often a source of confusion.

It is computationally convenient, and mathematically sensible, to label years with the sequence of integers $\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots$ so that year 0 precedes year 1; we do this when extrapolating backward on the Gregorian calendar and the same leap-year rule applies based on divisibility by 4, 100, and 400. However, on the Julian calendar it is customary to refer to the year preceding 1 C.E. as 1 B.C.E.,* counting it as a leap year in accordance with the every-fourth-year leap-year rule of the Julian calendar. Thus the epoch of the Hebrew calendar can alternatively be referred to as Saturday, October 5, 3761 B.C.E. (Julian). To highlight this asymmetry, in this paper we will append 'B.C.E.' *only* to Julian calendar years, reserving the minus sign for Gregorian calendar years. Thus for $n \geq 0$,

$$-n \text{ (Gregorian)} = (n+1) \text{ B.C.E. (Julian)},$$

and, for $n \geq 1$,

$$n \text{ (Gregorian)} = n \text{ C.E. (Gregorian)} = n \text{ C.E. (Julian)}.$$

*Before the common era; or, B.C.

Astronomers avoid this confusing situation by specifying dates in 'julian days'.[4, 5] Introduced in 1583 by Joseph Justus Scaliger, this method of dating is a strict counting of days backward and forward from

$$\begin{aligned}\text{J.D. } 0 &= \text{Monday, January 1, 4713 B.C.E. (Julian)}\\ &= \text{Monday, November 24, } -4711 \text{ (Gregorian).}\end{aligned}$$

Appendix A, section 1 of Neugebauer[4] explains the choice of starting date in detail. Using Scaliger's system, for example, the epoch of the Hebrew calendar is J.D. 347,996; the literature on the Mayan calendar commonly specifies epochs in julian days. Since our absolute date 0 is J.D. 1,721,425, it follows that

$$\text{J.D. } n = \text{our absolute date } (n - 1721425).$$

THE MAYAN CALENDAR

The Maya, developers of an ancient Amerindian civilization in Central America, employed three separate, overlapping calendrical systems, the *long count*, the *haab*, and the *tzolkin*.[6–13] Their civilization reached its zenith during the period 900–250 B.C.E., and although the Maya survive to this day in Guatemala and in the Yucatan peninsula of Mexico, the precise rules governing their calendars have been lost. What is known today has been recovered through 'astro-archeological' and historical research. There is general agreement on the Mayan calendrical rules; however, the exact correspondence between the Mayan calendars and Western calendars is still a matter of some dispute. Correspondences are proposed by interpreting Mayan recordings of astronomical phenomena, such as eclipses. In this paper, we give the details for the two most popular of the correspondences, the *Goodman–Martinez–Thompson correlation*[6] and *Spinden's correlation*.[11–13*] A superb discussion of Mayan mathematics, astronomy, and calendrical matters is given by Lounsbury.[14]

The long count is a strict counting of days from the beginning of the current cycle, each cycle containing 2,880,000 days (about 7885 solar years); the Maya believed that the universe is destroyed and re-created at the start of every cycle. The units of the long count are

1 kin	= 1 day[†]	
1 uinal	= 20 kin	(20 days)
1 tun	= 18 uinal	(360 days)
1 katun	= 20 tun	(7200 days)
1 baktun	= 20 katun	(144,000 days)

*Some of Spinden's date calculations are wrong. Here are three examples: on page 46 of 'Maya dates and what they reveal',[13] he gives the equivalence J.D. 1,785,384 = February 10, 176 (Gregorian), but it should be February 11, 176 (Gregorian); on top of page 55 several Gregorian dates are off by one day; on page 57 he gives the equivalence J.D. 2,104,772 = August 30, 1050 (Gregorian), but it should be July 27, 1050 (Gregorian).

†It is uncertain whether the Maya considered the day as beginning at sunset or midnight.

Thus, the long count date 12.16.11.16.6 means 12 baktun, 16 katun, 11 tun, 16 uinal, and 6 kin, for a total of 1,847,486 days from the start of the Mayan calendar epoch. We write Mayan long count dates as Lisp lists of the constituent units, for example, `(12 16 11 16 6)`.

Although not relevant here, the Maya used the following larger units for longer time periods:

1 pictun	= 20 baktun	(2,880,000 days)
1 calabtun	= 20 pictun	(57,600,000 days)
1 kinchiltun	= 20 calabtun	(1,152,000,000 days)
1 alautun	= 20 kinchiltun	(23,040,000,000 days)

An alautun is about 63,081,377 solar years!

The starting epoch of the long count, according to the Goodman–Martinez–Thompson correlation,[6] is taken as Wednesday, August 13, -3113 (Gregorian). This date equals September 8, 3114 B.C.E. (Julian), which equals J.D. 584,285, that is, day $-1,137,140$ in our absolute reckoning. In other words, our absolute date 0 is long count 7.17.18.13.0, which is 1,137,140 days after the Mayan calendar epoch. Since this value will be used several times, we define it as a Lisp constant,

```
(defconstant mayan-days-before-absolute-zero
;; Number of days of the Mayan calendar epoch before absolute day 0,
;; according to the Goodman-Martinez-Thompson correlation.
  1137140)
```

Spinden[11–13] proposes Monday, October 15, -3373 (Gregorian) as the epoch of the Mayan long count. This date equals November 11, 3374 B.C.E. (Julian), which equals J.D. 489,384, that is, day $-1,232,041$ in our absolute reckoning. By Spinden's epoch, our absolute date 0 is long count 8.11.2.6.1. To use Spinden's correlation we would instead define

```
(defconstant mayan-days-before-absolute-zero
;; Number of days of the Mayan calendar epoch before absolute day 0,
;; according to the Spinden correlation.
  1232041)
```

Thus, to convert from a Mayan long count date to an absolute date we need only compute the total number of days given by the long count and subtract the number of days before absolute date 0:

```
(defun absolute-from-mayan-long-count (count)
;; Absolute date corresponding to the Mayan long count count,
;; which is a list (baktun katun tun uinal kin).
  (+ (* (first count)  144000);; Baktun.
     (* (second count) 7200)  ;; Katun.
     (* (third count)  360)   ;; Tun.
     (* (fourth count) 20)    ;; Uinal.
     (fifth count)            ;; Kin (days).
     (-                       ;; Days before absolute date 0.
      mayan-days-before-absolute-zero)))
```

The COMMON LISP functions `first`, `second`, `third`, `fourth`, and `fifth` select, respectively, the first through fifth elements of a list.

In the opposite direction, converting an absolute date to a Mayan long count date, we need to add the number of days in the long count before absolute date 0, and then divide the result into baktun, katun, tun, uinal, and kin:

```
(defun mayan-long-count-from-absolute (date)
;; Mayan long count date of absolute date date.
  (let* ((long-count (+ date mayan-days-before-absolute-zero))
         (baktun (quotient long-count 144000))
         (day-of-baktun (mod long-count 144000))
         (katun (quotient day-of-baktun 7200))
         (day-of-katun (mod day-of-baktun 7200))
         (tun (quotient day-of-katun 360))
         (day-of-tun (mod day-of-katun 360))
         (uinal (quotient day-of-tun 20))
         (kin (mod day-of-tun 20)))
    (list baktun katun tun uinal kin)))
```

Here we have used `quotient`, a more aptly named call to COMMON LISP's two-argument version of `floor`,

```
(defun quotient (m n)
   (floor m n))
```

which returns the value $\lfloor m/n \rfloor$.

The Maya used a civil calendar, the haab, based approximately on the solar year, consisting of 18 'months' of 20 days each, together with 5 'monthless' days at the end. The months were called*

(1) Pop	(7) Yaxkin	(13) Mac
(2) Uo	(8) Mol	(14) Kankin
(3) Zip	(9) Chen	(15) Muan
(4) Zotz	(10) Yax	(16) Pax
(5) Tzec	(11) Zac	(17) Kayab
(6) Xul	(12) Ceh	(18) Cumku

The five monthless days were an unlucky period called *Uayeb.* Unlike Gregorian months, the days of the haab months begin at 0 and indicate the number of *elapsed days* in the current month. Thus, 0 Uo follows 19 Pop, and the fifth monthless day is followed by 0 Pop. This method of counting is used for years in the Hindu calendar discussed later in this paper.

Because the haab calendar accounts for only 365 days (as compared to the mean length of the solar tropical year, 365.2422 days), it is assumed the civil calendar slowly drifted with respect to the seasons. Mayan astronomers were aware of this

*The haab and tzolkin month names are transliterated from the Yucatan Mayan dialect. The Guatemalan Maya used slightly different names.

drift, having calculated the solar year to be 365.2420 days long.[15] (The Maya also took careful measurements of the lunar synodic month, which they calculated, quite accurately, to be equal to 29.53059 days.)

The long count date 0.0.0.0.0 is considered to be haab date 8 Cumku (there is no disagreement here in the various correlations), which we specify by

```
(defconstant mayan-haab-at-epoch '(8 18))
```

representing haab dates as pairs `(day month)`, where `day` and `month` are integers in the ranges 0 to 19 and 1 to 19, respectively. Thus we treat Uayeb as a defective nineteenth month and can convert an absolute date to a haab date by

```
(defun mayan-haab-from-absolute (date)
;; Mayan haab date of absolute date date.
  (let* ((long-count (+ date mayan-days-before-absolute-zero))
         (day-of-haab
          (mod (+ long-count
                  (first mayan-haab-at-epoch)
                  (* 20 (1- (second mayan-haab-at-epoch))))
               365))
         (day (mod day-of-haab 20))
         (month (1+ (quotient day-of-haab 20))))
    (list day month)))
```

It is not possible to convert a haab date to an absolute date, since, without a 'year', there is no unique corresponding absolute date. We can ask, though, for the number of days between two dates on the haab calendar—the calculation is elementary:

```
(defun mayan-haab-difference (date1 date2)
;; Number of days from Mayan haab date date1 to the next
;; occurrence of Mayan haab date date2.
  (mod (+ (* 20 (- (second date2) (second date1)))
          (- (first date2) (first date1)))
       365))
```

Note that Common Lisp's `mod`, unlike `rem`, *always returns a non-negative value for a positive divisor*; we use this property in the last function and frequently below. In the absence of such a modulus function, a conditional test could be used to map a remainder of -1 to $n-1$, and so on. The function `mod` also applies to non-integer values, so, for example, `(mod x 1)` returns the fractional part of `x`.

The function `mayan-haab-difference` can be used as follows to compute the absolute date of the Mayan haab date on or before a given absolute date:

```
(defun mayan-haab-on-or-before (haab date)
;; Absolute date of latest date on or before absolute date date
;; that is Mayan haab date haab.
   (- date
      (mod (- date
              (mayan-haab-difference
               (mayan-haab-from-absolute 0) haab))
           365)))
```

The third Mayan calendar, the tzolkin (or divine) calendar, was a religious calendar consisting of two cycles, a thirteen day count and a cycle of twenty names:

(1) Imix	(5) Chicchan	(9) Muluc	(13) Ben	(17) Caban
(2) Ik	(6) Cimi	(10) Oc	(14) Ix	(18) Etznab
(3) Akbal	(7) Manik	(11) Chuen	(15) Men	(19) Cauac
(4) Kan	(8) Lamat	(12) Eb	(16) Cib	(20) Ahau

Unlike the haab, the counts and names cycle *simultaneously*, so, for example, 13 Etznab precedes 1 Cauac which precedes 2 Ahau which precedes 3 Imix, and so on. Since 20 and 13 are relatively prime, this progression results in 260 unique dates, forming the divine year.

For the tzolkin calculations, and in the remainder of the paper, we will need a function like `mod`, but with its values adjusted so that the modulus of a multiple of the divisor is the divisor itself, rather than 0:

```
(defun adjusted-mod (m n)
;; Positive remainder of m/n with n instead of 0.
  (1+ (mod (1- m) n)))
```

The long count date 0.0.0.0.0 is taken to be tzolkin date 4 Ahau (the different correlations agree on this, too). Representing tzolkin dates as pairs of positive integers `(number name)`, where `number` and `name` are integers in the ranges 1 to 13 and 1 to 20, respectively, we specify

```
(defconstant mayan-tzolkin-at-epoch '(4 20))
```

As with the haab, we can convert from an absolute date to a tzolkin date with

```
(defun mayan-tzolkin-from-absolute (date)
;; Mayan tzolkin date of absolute date date.
  (let* ((long-count (+ date mayan-days-before-absolute-zero))
         (number
          (adjusted-mod (+ long-count
                           (first mayan-tzolkin-at-epoch))
                        13))
         (name
          (adjusted-mod (+ long-count
                           (second mayan-tzolkin-at-epoch))
                        20)))
    (list number name)))
```

Just as with the haab calendar, it is impossible to convert a tzolkin date to an absolute date. Unlike the haab calendar, however, because day numbers and day names cycle simultaneously, to calculate the number of days between two given tzolkin dates requires the solution to a pair of simultaneous linear congruences.[16] Suppose we want to know the number of days x from tzolkin date $(d_1\ n_1)$ until the next occurrence of tzolkin date $(d_2\ n_2)$. We must have

$$d_1 + x \equiv d_2 \pmod{13},$$

or, equivalently, $x = d_2 - d_1 + 13i$, for some integer i. Similarly, we must have

$$x \equiv n_2 - n_1 \pmod{20},$$

which becomes

$$d_2 - d_1 + 13i \equiv n_2 - n_1 \pmod{20}.$$

Hence we need to know the values of i satisfying

$$13i \equiv n_2 - n_1 - d_2 + d_1 \pmod{20}.$$

Multiplying each side by -3, the multiplicative inverse of 13 modulo 20, gives

$$i \equiv 3(d_2 - d_1 - n_2 + n_1) \pmod{20},$$

from which we conclude that

$$x = d_2 - d_1 + 13[3(d_2 - d_1 - n_2 + n_1) \bmod 20].$$

However, because we want the *next* occurrence of $(d_2\ n_2)$, we must guarantee that x is non-negative. Thus we write,

```
(defun mayan-tzolkin-difference (date1 date2)
;; Number of days from Mayan tzolkin date date1 to the next
;; occurrence of Mayan tzolkin date date2.
  (let* ((number-difference (- (first date2) (first date1)))
         (name-difference (- (second date2) (second date1))))
    (mod (+ number-difference
            (* 13 (mod (* 3 (- number-difference name-difference))
                       20)))
         260)))
```

As with the haab calendar, this function can be used to compute the absolute date of the Mayan tzolkin date on or before a given absolute date:

```
(defun mayan-tzolkin-on-or-before (tzolkin date)
;; Absolute date of latest date on or before absolute date date
;; that is Mayan tzolkin date tzolkin.
    (- date
       (mod (- date (mayan-tzolkin-difference
                     (mayan-tzolkin-from-absolute 0)
                     tzolkin))
            260)))
```

A popular way for the Maya to specify a date was to use the haab and tzolkin dates together, forming a cycle of the least common multiple of 365 and 260 days, 18980 days or approximately 52 solar years. This cycle is called a *calendar round*,[15, 17] and we can ask for the most recent absolute date, on or before a given absolute date, that falls on a specified date of the calendar round.

Suppose the haab date of interest is $(D\ M)$ and the tzolkin date of interest is $(d\ n)$; we seek the latest absolute date x, on or before the given absolute date, that satisfies

$$\begin{aligned} x + D_0 &\equiv D \pmod{20}, \\ x + M_0 &\equiv M \pmod{18}, \\ x + d_0 &\equiv d \pmod{13}, \\ x + n_0 &\equiv n \pmod{20}, \end{aligned}$$

where $(D_0\ M_0)$ is the haab date of absolute date 0, and $(d_0\ n_0)$ is the tzolkin date of absolute date 0. The first two of these congruences combine to become

$$x \equiv \Delta_H \pmod{365}, \tag{1}$$

where

$\Delta_H =$ `mayan-haab-difference` applied to $(D\ M)$ and $(D_0\ M_0)$.

The last two of these congruences combine to become

$$x \equiv \Delta_T \pmod{260}, \tag{2}$$

where

$\Delta_T =$ `mayan-tzolkin-difference` applied to $(d\ n)$ and $(d_0\ n_0)$.

Congruence (1) means that

$$x = \Delta_H + 365i, \tag{3}$$

and combining this with congruence (2) we get

$$365i \equiv \Delta_T - \Delta_H \pmod{260}.$$

This has a solution *only* if $\Delta_T - \Delta_H$ is divisible by 5; otherwise the haab-tzolkin combination cannot occur. So, dividing by 5 we get

$$73i \equiv \frac{\Delta_T - \Delta_H}{5} \pmod{52},$$

and multiplying this by the multiplicative inverse of 73 modulo 52 (which, by coincidence, is 5), we find

$$i \equiv \Delta_T - \Delta_H \pmod{52},$$

so that $i = \Delta_T - \Delta_H + 52u$, for some u. Plugging this into equation (3) yields

$$x = \Delta_H + 365(\Delta_T - \Delta_H) + 18980u.$$

Thus we want the last date on or before the given absolute date that is congruent to $\Delta_H + 365(\Delta_T - \Delta_H)$ modulo 18980. This is computed in a manner identical to that in the function `Kday-on-or-before` in our earlier paper,[1] and we have:

```
(defun mayan-haab-tzolkin-on-or-before (haab tzolkin date)
;; Absolute date of latest date on or before date that is Mayan
;; haab date haab and tzolkin date tzolkin; returns nil if such
;; a haab-tzolkin combination is impossible.
  (let* ((haab-difference
          (mayan-haab-difference (mayan-haab-from-absolute 0)
                                 haab))
         (tzolkin-difference
          (mayan-tzolkin-difference (mayan-tzolkin-from-absolute 0)
                                    tzolkin))
         (difference (- tzolkin-difference haab-difference)))
    (if (= (mod difference 5) 0)
        (- date
           (mod (- date
                   (+ haab-difference (* 365 difference)))
                18980))
      nil)));;  haab-tzolkin combination is impossible.
```

This function can be used to compute the number of days between a pair of dates on the calendar round or to write a function `mayan-haab-tzolkin-on-or-after`; we leave these to the reader.

THE FRENCH REVOLUTIONARY CALENDAR

The French Revolutionary Calendar[9, 18] was instituted by the National Convention of the French Republic in October 1793. Its epoch was absolute date 654,415, that is, Saturday, September 22, 1792 (Gregorian), the autumnal equinox of that year, and also the first day following the establishment of the Republic. It went into effect on Sunday, November 24, 1793 (Gregorian), and was used by the French until Tuesday, December 31, 1805 (Gregorian); on Wednesday, January 1, 1806 (Gregorian), the Revolutionary calendar was abandoned by Napoleonic edict and France reverted to the Gregorian calendar.

Following the example of several ancient calendars, including the Coptic, Ethiopic, and Persian,[9] the French Revolutionary calendar divided the year into 12 months containing exactly 30 days each, followed by a period of five monthless days (six in leap years). The names of the twelve months, poetically coined by Fabre d'Eglantine, were taken from the seasons in which they occurred:

(1) Vendémiaire (vintage)	(7) Germinal (seed)
(2) Brumaire (fog)	(8) Floréal (blossom)
(3) Frimaire (sleet)	(9) Prairial (pasture)
(4) Nivôse (snow)	(10) Messidor (harvest)
(5) Pluviôse (rain)	(11) Thermidor (heat)
(6) Ventôse (wind)	(12) Fructidor (fruit)

Contemporary British journalists sarcastically dubbed them Slippy, Nippy, Drippy, Freezy, Wheezy, Sneezy, Showery, Flowery, Bowery, Wheaty, Heaty, Sweety.

We will use a list `(month day year)` to represent the date, treating the monthless days as a thirteenth month, as in the Mayan haab calendar:

```
(defun french-last-day-of-month (month year)
;; Last day of month, year on the French Revolutionary calendar.
  (if (< month 13)
      30
    (if (french-leap-year year)
        6
      5)))
```

The leap-year rules are given below.

Although not relevant to our calculations, each month was divided into three 'decades' of ten days each; the tenth day was considered a day of rest. (This made the new calendar unpopular because under the Gregorian calendar the workers had had every seventh day off.) The ten days were named by their ordinal position in the decade,

(1) Primidi	(6) Sextidi
(2) Doudi	(7) Septidi
(3) Tridi	(8) Oxtidi
(4) Quartidi	(9) Nonidi
(5) Quintidi	(10) Decadi

Each day was divided into ten 'hours', each of which was divided into one hundred 'minutes', each of which was divided into one hundred 'seconds'.

The five or six monthless days that were added at the end of each year were holidays called *sansculottides*, celebrating various attributes of the Revolution:

(1) Jour de la Vertu (virtue day)
(2) Jour du Genie (genius day)
(3) Jour du Labour (labor day)
(4) Jour de la Raison (reason day)
(5) Jour de la Recompense (reward day)
{(6) Jour de la Revolution (revolution day)}

The leap-year structure is given in curly brackets.

The leap year rule is complicated by historical fact.[18] Originally, the calendar was to be kept in synchronization with the solar year by using the extra day in a leap year to force the new year, 1 Vendémiaire, to occur at the autumnal equinox. Unfortunately, the irregular variations of a day or two in the occurrence of the equinox made this impracticable and a proposal was put forward for a simpler, more regular, leap year rule; however, the calendar was abandoned before this rule could be adopted. Thus, leap years *actually occurred* in the 3rd, 7th, and 11th years and *would have occurred* in the 15th and 20th years, had the calendar still been in use. The proposed rule then would have been:

every 4th year is a leap year, except
every 100th year is not a leap year, except
every 400th year is a leap year, except
every 4000th year is not a leap year,

giving an average of 365.24225 days per year, an error of about 1 day in 20,000 years.

The following Lisp function uses historical practice based on equinoxes for years 1 through 20 and uses the proposed rule for years beyond 20:

```
(defun french-leap-year (year)
;; True if year is a leap year on the French Revolutionary calendar.
  (or (member year '(3 7 11));; Actual.
      (member year '(15 20)) ;; Anticipated.
      (and (> year 20)       ;; Proposed.
           (= 0 (mod year 4))
           (not (member (mod year 400) '(100 200 300)))
           (not (= 0 (mod year 4000))))))
```

Conversion of a French Revolutionary date to an absolute date is thus done by summing all days before that date, including the number of days before the calendar began, 365 days for each prior year, all prior leap days, and the number of prior days in the present year:

```
(defun absolute-from-french (date)
;; Absolute date of French Revolutionary date.
  (let* ((month (first date))
         (day (second date))
         (year (third date)))
    (+ 654414;; Days before start of calendar.
       (* 365 (1- year));; Days in prior years.
       ;; Leap days in prior years.
       (if (< year 20)
           (quotient year 4);; Actual and anticipated practice,
                            ;; that is, years 3, 7, 11, and 15.
         ;; Proposed rule--there were 4 leap years before year 20.
         (+ (quotient (1- year) 4)
            (- (quotient (1- year) 100))
            (quotient (1- year) 400)
            (- (quotient (1- year) 4000))))
       (* 30 (1- month));; Days in prior months this year.
       day)));; Days so far this month.
```

Calculating the French Revolutionary date from the absolute date d involves sequentially determining the year, month, and day of the month. The year is first closely approximated from below by $\lfloor d/366 \rfloor$ and then found precisely by stepping through subsequent years (that is, by a linear search). The month is then found by a similar linear process, and the day of the month is determined by subtraction:

```
(defun french-from-absolute (date)
;; French Revolutionary date (month day year) of absolute date;
;; returns nil if date is before the French Revolution.
  (if (< date 654415)
      nil;; pre-French Revolutionary date.
    (let* ((approx      ;; Approximate year from below.
```

```
               (quotient (- date 654414) 366))
              (year        ;; Search forward from the approximation.
               (+ approx
                  (sum 1 y approx
                    (>= date
                        (absolute-from-french (list 1 1 (1+ y)))))))
              (month       ;; Search forward from Vendemiaire.
               (1+ (sum 1 m 1
                     (> date
                        (absolute-from-french
                         (list m
                               (french-last-day-of-month m year)
                               year))))))
              (day         ;; Calculate the day by subtraction.
               (- date
                  (1- (absolute-from-french (list month 1 year))))))
       (list month day year))))
```

THE OLD HINDU CALENDARS

The Hindus have both solar and lunar calendars. Solar dates are in use primarily in southern India and lunar dates are used elsewhere; virtually all religious holidays are determined by the lunar calendar. In the Hindu lunar system, like other lunisolar calendars, months follow the lunar cycle and are synchronized with the solar year by introducing occasional leap months. Unlike the lunisolar Hebrew calendar (described in algorithmic detail in our previous paper[1]), the Hindu intercalated months do not follow a fixed pattern; instead, like the lunisolar Chinese calendar,[19] their occurrence depends on astronomical factors. However, unlike other calendars, a day can be *omitted* any time in the lunar month.

Current Hindu calendars are based on approximations to the *true* times of the sun's entrance into the signs of the zodiac and of lunar conjunctions (new moons). But, some earlier Hindu calendars appear instead to have used an approximation to *mean* times.[20, 21] It is this mean calendar, as described in sections 6–15 of van Wijk,[20] that we implement here. The mean and true calendars can differ by a few days and/or can be shifted by a month. For an ancient description of Hindu astronomy, calendars, and holidays, see the book on India by al-Biruni.[22]*

There are various epochs that are, or have been, used as starting points for the enumeration of years in India, and for each there are two versions, one beginning with year 0 (expired years) and the other with year 1 (current year). We use the *Kali Yuga* ('Iron Age') epoch, according to which the beginning of the first day of year 0 K.Y. is midnight at the start of Friday, January 23, −3101 (Gregorian). This

*There is some confusion of dates in the note (on page 358) by R. Schram to volume II, page 2 of Sachau's translation of this book, where the following equivalences are given: Thursday, February 25, 1031 C.E. (Julian) = 1 Caitra 953 Śaka Era = 28 Ṣafar 422 Anno Hijrae = 19 Ispandârmadh-Mâh 399 Anno Persarum, and New Year 400 Anno Persarum = March 9, 1031 C.E. (Julian) = J.D. 2,097,686. In fact, February 25, 1031 C.E. (Julian) = 29 Ṣafar 422 Anno Hijrae = J.D. 2,097,686.

date equals February 18, 3102 B.C.E. (Julian), which equals J.D. 588,466, that is, day $-1, 132, 959$ in our absolute reckoning. That midnight is considered to have been the start of a new solar year and a new lunar month; indeed, according to the data in the *Sūrya-Siddhānta*,[23] it is the time of the most recent conjunction of all the visible planets. The (expired) year number is the number of solar (sidereal, not tropical*) years that have *elapsed* since the onset of the Kali Yuga. As van Wijk[20] explains:

> We count the years of human life in expired years. A child of seven years has already lived more than seven years; but on the famous *18 Brumaire de l'An VIII de la République Française une et indivisible* only 7 years and 47 days of the French Era had elapsed.

The Kali Yuga epoch marks—in Hindu chronology—the onset of the fourth and final stage (lasting 432,000 years) of the 4,320,000-year era beginning with the last re-creation of the world.

The solar months are named after the signs of the zodiac (but may differ from region to region):

(1) Mesha	(Aries)	(7) Tulā	(Libra)
(2) Vṛshabha	(Taurus)	(8) Vṛiśchika	(Scorpio)
(3) Mithuna	(Gemini)	(9) Dhanus	(Sagittarius)
(4) Karka	(Cancer)	(10) Makara	(Capricorn)
(5) Simha	(Leo)	(11) Kumbha	(Aquarius)
(6) Kanyā	(Virgo)	(12) Mīna	(Pisces)

The (sidereal) solar year is taken to be

$$\frac{1,577,917,828}{4,320,000} = 365.25875648148148\cdots$$

(civil) days long. Different Indian astronomical treatises give slightly varying constants; this, and the constants that follow, are from the *Sūrya-Siddhānta*,[23] circa 1000 C.E. The solar new year is called *Mesha saṃkrānti*; it is the day of the first sunrise after the sun enters Mesha. Each subsequent month begins when the sun has traveled one twelfth of the way around the ecliptic (the sun's path in the sky) and entered a new sign of the zodiac. If the sign is entered before sunrise, then it is day one of a new month; otherwise, it is the last day of the previous month. Hence, even though in the mean system the speed of the sun is taken to be constant, months vary in length.

The name of a lunar month depends on the position of the sun in the zodiac at the beginning of that lunar month, and is derived from the names of asterisms (star groups) along the ecliptic:

*A *tropical year* is the time it takes for the sun to return to the same position in its apparent path, while a *sidereal year* is the time it takes for the sun to return to the same celestial longitude; the difference is slight.

(1) Chaitra	(7) Āśvina
(2) Vaiśākha	(8) Kārttika
(3) Jyaishṭha	(9) Mārgaśīra
(4) Āshāḍha	(10) Pausha
(5) Śrāvaṇa	(11) Māgha
(6) Bhādrapada	(12) Phālguna

Some regions of India begin the year with Kārttika and/or use different or shifted month names. We follow the south-India method in which months begin and end at new moons (the *amânta* scheme); in the north, months go from full moon to full moon (the *pûrṇimânta* scheme). Our absolute day 0 is 18 Makara, 3101 K.Y. on the mean solar calendar and 19 Pausha on the lunar one.

Since a solar month (the mean time for the sun to traverse a zodiacal sign) is longer than a lunar month, a lunar month is intercalated whenever two new moons occur within one sign of the Hindu zodiac. The two months starting within that sign take the same name, except that the first is called *adhika* ('added') and the second *nija* ('regular').

The lunar cycle is divided into thirty phases (or 'lunar days'), called *tithis*; because a mean lunar month is less than thirty (civil) days long, a tithi is somewhat shorter than a day. The first fifteen tithis belong to the *suddha* ('bright', waxing) fortnight and the second fifteen, to the *bahula* ('dark', waning) fortnight. The day of the lunar month corresponds to the phase of the moon at sunrise, and is usually referred to by ordinal number within one fortnight or the other; we use ordinal numbers from 1 to 30. Just as there are leap months, caused by two new moons occurring within the same zodiacal sign, there are also 'lost' days whenever a tithi begins and ends between one sunrise and the next.

Suppose we can determine the longitudinal positions of the sun and moon, relative to the celestial sphere, at any given time. Then to determine the lunar Hindu date of any given day, we perform the following sequence of operations:

1. Determine the phase of the moon at sunrise of the given day, by taking the difference in longitudes between the positions of the sun and moon. This gives the ordinal number of the tithi current at sunrise.

2. Determine when the last new moon was before sunrise of the current day.

3. Determine the position of the sun at that new moon. The zodiacal sign in which it was determines the name of the current month.

4. Compare the current month name with that of the next new moon. If they are the same, then it is a leap month.

The calendars in use in recent times follow a similar pattern, but require much more elaborate calculation to approximate the true positions of the sun and moon and the time of local sunrise. These more elaborate calculations can result, on rare occasions, in expunged months and intercalated days.[20, 24]

Four constants play a role in the mean computations:

```
(defconstant solar-sidereal-year (+ 365 279457/1080000))
(defconstant solar-month (/ solar-sidereal-year 12))
(defconstant lunar-sidereal-month (+ 27 4644439/14438334))
(defconstant lunar-synodic-month (+ 29 7087771/13358334))
```

We work exclusively with rational numbers; otherwise, double precision is required for many of the calculations. Unlike table-based methods, using rational numbers gives perfect fidelity to the sources.

The sidereal year is the mean number of days it takes for the sun to return to the same longitudinal point with respect to the stars, and the solar month is the mean time it takes it to traverse one sign of the zodiac. Similarly, the sidereal month is the mean time it takes for the moon to return to the same point, in which time the sun would have moved farther on. The synodic month takes the motion of the sun into account—it is the mean time between new moons (lunar conjunctions). Civil days begin at sunrise; we use mean sunrise, one quarter of a day past midnight, that is, 6:00A.M., in our calculations.

Once we know the position of the sun in degrees of celestial longitude,

```
(defun solar-longitude (days)
;; Mean sidereal longitude of the sun, in degrees,
;; at date and fraction of day days.
  (* (mod (/ days solar-sidereal-year) 1) 360))
```

we can determine the zodiacal sign:

```
(defun zodiac (days)
;; Zodiacal sign of the sun, as integer in range 1..12,
;; for date and fraction of day days.
  (1+ (quotient (solar-longitude days) 30)))
```

Now, converting an absolute date to the date according to the Hindu solar calendar is straightforward:

```
(defun old-hindu-solar-from-absolute (date)
;; Hindu solar month, day, and year of absolute date date.
  (let* ((hdate (+ date 1132959 1/4));;   Sunrise on Hindu date.
         (year (quotient hdate solar-sidereal-year))
         (month (zodiac hdate))
         (day (1+ (floor (mod hdate solar-month)))))
    (list month day year)))
```

Inverting this calculation is no harder:

```
(defun absolute-from-old-hindu-solar (date)
;; Absolute date corresponding to Hindu solar date date.
  (let* ((month (first date))
         (day (second date))
         (year (third date)))
    (floor (+ (* year solar-sidereal-year);; Days in elapsed years.
              (* (1- month) solar-month)  ;; In months.
              day -1/4                    ;; Whole days until midnight.
              -1132959))))                ;; Days before absolute day 0.
```

To compute the lunar date, we need to determine the position of the moon in degrees of celestial longitude:

```
(defun lunar-longitude (days)
;; Mean sidereal longitude of the moon, in degrees,
;; at date and fraction of day days.
  (* (mod (/ days lunar-sidereal-month) 1) 360))
```

To get the phase of the moon, we must measure the distance between the positions of the sun and the moon. The following function converts that value, in degrees, to a number between 1 and 30:

```
(defun lunar-phase (days)
;; Longitudinal distance between the sun and the moon, as an integer
;; in the range 1..30, at date and fraction of day days.
  (1+ (quotient
       (mod (- (lunar-longitude days) (solar-longitude days))
            360)
       12)))
```

To determine the number of the month, we look back to the last occurrence of a new moon, see where the sun was at that time, and then give the lunar month the number of the next solar month. Similarly, for the lunar year number, we use the solar year in which the next lunar month falls. The most recent new moon is found by

```
(defun new-moon (days)
;; Time of the most recent mean conjunction at or before
;; date and fraction of day days.
  (- days (mod days lunar-synodic-month)))
```

We are now ready to compute the Hindu date, given any absolute date. We use a list `(month leapmonth day year)` to represent a date, where `month` is an integer in the range 1 through 12, `day` is an integer in the range 1 through 30, `year` is a non-negative integer, and `leapmonth` is either `t` or `nil`, corresponding to 'true' and 'false':

```
(defun old-hindu-lunar-from-absolute (date)
;; Hindu lunar month, day, and year of absolute date date.
  (let* ((hdate (+ date 1132959))      ;; Hindu date.
         (sunrise (+ hdate 1/4))       ;; Sunrise on that day.
         (last-new-moon                ;; Last new moon.
          (new-moon sunrise))
         (next-new-moon                ;; Next new moon.
          (+ last-new-moon lunar-synodic-month))
         (day (lunar-phase sunrise))   ;; Day of month.
         (month                        ;; Month of lunar year.
          (adjusted-mod (1+ (zodiac last-new-moon)) 12))
         (leapmonth                    ;; If next month the same.
```

```
          (= (zodiac last-new-moon)
             (zodiac next-new-moon)))
         (next-month                   ;; Beginning of next month.
          (+ next-new-moon
                        (if leapmonth lunar-synodic-month 0)))
         (year                         ;; Solar year of next month.
          (quotient next-month solar-sidereal-year)))
    (list month leapmonth day year)))
```

Unlike the French Revolutionary calendar, or the calendars in our previous paper,[1] there is no direct way of converting a lunar Hindu date to an absolute date, since years and months do not have a fixed length. To invert the process and derive the absolute date from a Hindu lunar date, we first find a lower bound on the possible absolute date, and then perform a linear search for the exact correspondence. For this purpose, it is convenient to be able to compare Hindu lunar dates:

```
(defun old-hindu-lunar-precedes (date1 date2)
;; True if Hindu lunar date1 precedes date2.
  (let* ((month1 (first date1))
         (month2 (first date2))
         (leap1 (second date1))
         (leap2 (second date2))
         (day1 (third date1))
         (day2 (third date2))
         (year1 (fourth date1))
         (year2 (fourth date2)))
    (or (< year1 year2)
        (and (= year1 year2)
             (or (< month1 month2)
                 (and (= month1 month2)
                      (or (and leap1 (not leap2))
                          (and (equal leap1 leap2)
                               (< day1 day2)))))))))
```

Finally, taking into account the possible nonexistence of a particular date,

```
(defun absolute-from-old-hindu-lunar (date)
;; Absolute date corresponding to Hindu lunar date date;
;; returns nil if no such date exists.
  (let* ((years (fourth date))      ;; Elapsed years.
         (months (- (first date) 2));; Elapsed whole months,
               ;; minus a month's possible difference between the
               ;; solar and lunar year.
         (approx;;  Approximate date from below by adding days...
          (+ (floor (* years solar-sidereal-year)) ;;  in years,
             (floor (* months lunar-synodic-month));;  in months,
             -1132959))            ;; and before absolute date 0.
         (try
          (+ approx             ;; Search forward to correct date,
             (sum 1 i approx    ;; or just past it.
```

```
                   (old-hindu-lunar-precedes
                    (old-hindu-lunar-from-absolute i)
                    date)))))
     (if (equal (old-hindu-lunar-from-absolute try) date)
         try
       nil)));; date non-existent on Hindu lunar calendar.
```

CONCLUSION

Our goal in this series of papers is to make accurate calendrical algorithms readily available, since calendrical problems are notorious for plaguing software. Some examples:

1. The COBOL programming language allocates only two decimal digits for internal storage of years, so untold numbers of programs are expected to go awry on New Year's Eve of the coming century.[25]

2. Many programs err in, or simply ignore, the century rule for leap years on the Gregorian calendar (every 4th year is a leap year, except every 100th year which is not, except every 400th year which is). For example, early releases of the popular spreadsheet program Lotus® 1-2-3® treated 2000 as a non-leap year, a problem eventually fixed. But, all releases of Lotus® 1-2-3® take 1900 as a leap year; by the time this error was recognized, the company deemed it too late to correct: 'The decision was made at some point that a change now would disrupt formulas which were written to accomodate [*sic*] this anomaly.'[26]

3. Various programs calculate the Hebrew calendar by first determining the date of Passover using Gauss's method;[27] this method is correct only when sufficient precision is used, so such an approach often leads to errors.

4. At least one modern, standard source for calendrical matters, Parise,[9] has errors, some of which are presumably due to the algorithms used to produce the tables.*

Though the calendars described in this paper are mainly of historical interest, the Mayan and French Revolutionary calendars are incorporated in version 19 of GNU Emacs.[28] The Hindu can serve as the basis for implementing the various more intricate calendars in use on the Indian subcontinent today, and perhaps also the Chinese calendar.[19] The algorithms presented also serve to illustrate some basic features of nonstandard calendars: The Mayan calendar requires dealing with multiple, independent cycles and exemplifies the kind of reasoning often needed for calendrical-historical research. The French calendar is an example of one in which two cycles (years and 'decades') are synchronized. The Hindu calendar is an example of one in which the cycles (days of the month, months of the year) are irregular.

*Examples include: The Mayan date 8.1.19.0.0 is given incorrectly as February 14, 80 (Gregorian) on page 290; the dates given on pages 325–327 for Easter for the years 1116, 1152, and 1582 are not Sundays; the epact given for 1986 on page 354 is wrongly given as 20.

ACKNOWLEDGEMENTS

The second author's research was supported in part by the U. S. National Science Foundation under Grants CCR-90-07195 and CCR-90-24271 and by a Lady Davis fellowship at the Hebrew University in Jerusalem.

REFERENCES

Note: Only English language references are included in the following list. Some of these are relatively inaccessible.

1. N. Dershowitz and E. M. Reingold, 'Calendrical calculations', *Software—Practice and Experience*, **20**, (9), 899–928 (Sept. 1990).
2. Al-Biruni (= Abû-Raiḥân Muḥammad ibn 'Aḥmad al-Bîrûnî), *Alâthâr Albâḳiya 'an-il-Kûrûn Alkhâliya*, 1000. Translated and annotated by C. E. Sachau as *The Chronology of Ancient Nations*, William H. Allen and Co., London, 1879; reprinted by Hijra International Publishers, Lahore, Pakistan, 1983.
3. G. L. Steele, Jr., COMMON LISP: *The Language*, 2nd edn., Digital Press, Bedford, MA, 1990.
4. O. Neugebauer, *A History of Ancient Mathematical Astronomy*, Springer-Verlag, Berlin, 1975.
5. *Explanatory Supplement to the Astronomical Ephemeris and the American Ephemeris and Nautical Almanac*, Her Majesty's Stationery Office, London, 1961.
6. J. E. S. Thompson, *Maya Hieroglyphic Writing*, 3rd edn., University of Oklahoma Press, Norman, OK, 1971.
7. L. Satterwaite, 'Concepts and structures of Maya calendrical arithmetics', *Ph.D. Thesis*, University of Pennsylvania, Philadelphia, 1947.
8. B. Tedlock, *Time and the Highland Maya*, University of New Mexico Press, Albuquerque, 1982.
9. F. Parise, *The Book of Calendars*, Facts on File, New York, 1982.
10. J. Hastings, ed., *Encyclopædia of Religion and Ethics*, Charles Scribner's Sons, New York, 1911.
11. H. J. Spinden, 'Central American calendars and the Gregorian day', *Proc. Nat. Acad. Sci.* (*USA*), **6**, 56–59 (1920).
12. H. J. Spinden, 'The reduction of Maya dates', *Peabody Museum Papers*, **VI**, (4), (1924).
13. H. J. Spinden, 'Maya dates and what they reveal', *Science Bulletin* (The Museum of the Brooklyn Institute of Arts and Sciences), **IV**, (1), (1930).
14. F. G. Lounsbury, 'Maya numeration, computation, and calendrical astronomy', *Dictionary of Scientific Bibliography*, **15**, Supplement 1, 759–818, Charles Scribner's Sons, New York, 1978.
15. S. G. Morley, *The Ancient Maya*, revised by G. W. Brainerd, Stanford University Press, Stanford, CA, 1963.
16. O. Ore, *Number Theory and Its History*, McGraw-Hill Book Co., Inc., New York, 1948. Reprinted by Dover Publications, Inc., Mineola, NY, 1987.
17. C. P. Bowditch, *The Numeration, Calendar Systems and Astronomical Knowledge of the Mayas*, Cambridge University Press, Cambridge, 1910.
18. M. Hamer, 'A calendar for all seasons', *New Scientist*, **124**, (1696/1697), 9–12 (Dec. 23/30, 1989).
19. H. Fritsche, *On Chronology and the Construction of the Calendar with Special Regard to the Chinese Computation of Time Compared with the European*, R. Laverentz, St. Petersburg, 1886.
20. W. E. van Wijk, *Decimal Tables for the Reduction of Hindu Dates from the Data of the Sūrya-Siddhānta*, Martinus Nijhoff, The Hague, 1938.

21. H. Jacobi, 'The computation of Hindu dates in inscriptions, &c.', *Epigraphia Indica: A Collection of Inscriptions Supplementary to the Corpus Inscriptionum Indicarum of the Archæological Survey of India*, J. Burgess, ed., Calcutta, 1892, 403–460, 481.
22. Al-Biruni (= Abû-Raiḥân Muḥammad ibn 'Aḥmad al-Bîrûnî), *India: An accurate description of all categories of Hindu thought, as well those which are admissible as those which must be rejected*, circa 1030. Translated and annotated by E. C. Sachau, *Alberûnî's India: An account of the religion, philosophy, literature, geography, chronology, astronomy, customs, laws and astrology of India*, William H. Allen and Co., London, 1910; reprinted under the Authority of the Government of West Pakistan, Lahore, 1962 and by S. Chand & Co., New Delhi, 1964.
23. *Sūrya-Siddhānta*, circa 1000. Translated by E. Burgess with notes by W. D. Whitney, *J. Amer. Oriental Soc.*, New Haven, 1860. A new edition, edited by P. Gangooly with an introduction by P. Sengupta, was published by Calcutta University, 1935. This edition was reprinted by Indological Book House, Varanasi, India, 1977.
24. V. B. Ketkar, 'Indian and foreign chronology: with theory, practice and tables, B.C. 3102 to 2100 A.D. and notices of the Vedic, the ancient Indian, the Chinese, the Jewish, the ecclesiastical and the Coptic calendars', *J. Royal Asiatic Soc., Bombay Branch*, **XXVI**, (LXXV-A, *Extra Number*), 1923.
25. P. G. Neumann, 'Inside risks: the clock grows at midnight', *Comm. ACM*, **34**, (1), 170 (Jan., 1991).
26. Letter to Nachum Dershowitz from Kay Wilkins, Customer Relations Representative, Lotus Development Corporation, Cambridge, MA, April 21, 1992.
27. I. Rhodes, 'Computation of the dates of the Hebrew New Year and Passover', *Comp. & Maths. with Appls.*, **3**, 183–190 (1977).
28. R. M. Stallman, *GNU Emacs Manual*, 6th edn., Free Software Foundation, Cambridge, MA, 1986.