```
Shire Calendar

12 months of 30 days
year starts on (Saturday) Sterrendei  with 2 Yule

month names = { Afteryule, Solmath, Rethe, Astron, Thrimidge, Forelithe,
                Afterlithe, Wedmath, Halimath, Winterfilth, Blotmath, Foreyule}
Days names = {Sterrendei, Sunnendei, Momendei, Trewesdei, Hevenesdei, Meresdei, Highdei }

Year =>2Yule, Afteryule, Solmath, Rethe, Astron, Thrimidge, Forelithe, 1Lithe,
        Midyear's Day, (Overlithe),     [Not days of any week, overlithe for leap year]
        2Lithe, Afterlithe, Wedmath, Halimath, Winterfilth, Blotmath, Foreyule, 1Yule
*******************************************************************************
Gregorian Calendar
12 months of varying days and various starting days

month names = { January, February, March, April, May, June, July, August, September,
                October, November, December }
days names = {Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday}
*******************************************************************************
???? Calendar
13 months of 28 days with MidDay in the seventh month between 14 and 15 and on
leap years LeapDay by midday
*******************************************************************************

Years divisible by 4 a leap year except divisible by 100 unless divisible by 400

*******************************************************************************
Internal date:time format is Julian GMT -  year:day:sssss.sss
time zones internal ±sssss

sub time2zulu { }
This subroutine converts a zone julian date:time to zulu given original date:time and time zone
Algorithm:
    add time to original
    if time > 86400 then subtract 86400 from time and increment date
    if time < 0000 then add 2400 to time and decrement date
    if day > year's end then date=1 and increment year
    if day < 1 then day=1 and decrement year

sub zulu2time { }
This subroutine converts a zulu date:time to zone given zulu date:time and time zone
Algorithm:
    subtract time from zulu
    if time > 2400 then subtract 2400 from time and increment date
    if time < 0000 then add 2400 to time and decrement date
    if day > year's end then date=1 and increment year
    if day < 1 then day=1 and decrement year

sub Gregorian2Julian { }
This subroutine converts a Gregorian date to a Julian date
Algorithm:
    Determine if leap year and use appropriate month set
    determine integer month
    return day = day of month plus sum of previous months

sub Julian2Gregorian { }
This subroutine converts a Julian date to a Gregorian date
Algorithm:
    Determine if leap year and use appropriate month set
    While day !inMonth increment month
    Return Gregorian Date

sub Shire2Julian { }
```

```
This subroutine converts a Shire date to a Julian date
Algorithm:
    If special day then set value and return day
    Determine if leap year and use appropriate year set
    determine integer month
    return day = day of month plus sum of previous months

sub Julian2Shire { }
This subroutine converts a Julian date to a Shire date
Algorithm:
    Determine if leap year and use appropriate year set
    If special day then return Shire date by table
    While day !inMonth increment Month
    return Shire date

sub ShireDayofWeek { }
This subroutine returns the number of the day of week 0:7 given a Shire date
Algorithm:

sub GregorianDayofWeek { }
This subroutine returns the number of the day of week 1:7 given a Gregorian Date
Algorithm:

sub isLeap { }
This returns True if LeapYear otherwise False
Algorithm:
    if year mod 4 ≠ 0 then return False
    if year mod 4 = 0 and year mod 400 = 0 return True
    if year mod 4 = 0 and year mod 100 ≠ 0 return True else False

sub time2secs { }
This subroutine converts 24 hour time to seconds from midnight
Algorithm:

sub secs2time ( )
This subroutine converts seconds from midnight to 24 hour time
Algorithm:
    ss.sss = secs mod 60
    hour = trunc(secs - ss.sss)/3600
    minutes = ((secs - ss.sss) - (hour*3600))/60

sub 12convert24
This sub converts a 24 hour time to 12 hour time
Algorithm:
    if hh < 13 return time + 'a.m.'
        else hh -= 12 and return time + 'p.m.'

sub 24convert12
This sub converts a 12 hour time to 24 hour time
Algorithm:
    if 'a.m.' return time else hh += 12 and return time

sub deltaDates
This subroutine determines the delta time between two dates
Algorithm
    Determine the greater date
    subtract from the greater date the lesser date
        subtract seconds and make necessary day corrections
        subtract days and make any necessarey year corrections
        subtract years
    return delta time

sub dateDelta
This subroutine computes the new date given the original date and delta time
```

```
Algorithm
    add delta time to date making corrections as necessary
    return new date
```