```
!
! Source code for implementation of positive whole numbers in minimal 1's Standard Format from base-10 to any real number radix
!
!
! Copyright 2011, 2012, Colin James III   All Rights Reserved
!
!
!  Important note:  The "minimal" 1's Standard Format requires that two one's ("11") cannot be consecutive.
!
!                   For integer arithmetic, this causes problems of precision and rounding when testing integer parts
!                   that are slightly less than exact integers.
!
!                   The solution is in rounding to nine decimal places as applied only the mixed number result of the
!                   expression LOG( n) / log_phi.  The ROUND function is invoked once per iteration for a power of phi
!                   that is directly found.  The other operations for each iteration are one each of division, exponent,
!                   integer, logarithm, and subtraction.
!
!                   Integer arithmetic is used where the standard INT( n) is equivalent to FLOOR( n), such that 1.2 = 1,
!                   0.1 = 0, -0.1 = -1, and -1.2 = -2.
!
!                   The logarithmic radix chosen to convert numbers into base-Phi is the log base-e, the default of True BASIC.
!                   Base-e was tested as faster than, in order, base-10 and base-2.
!
!                   The user may specify other radix bases by changing in SUB Set_initial_values the value of phi2 to an
!                   expression.  Before the infinite processing loop begins, the user is prompted for a numeric value (not
!                   a string expression) as the radix base for that session.  The default radix is Phi.  Fractional radices
!                   other than Phi, such as Exp( 1) = e and pi, do not seem to work well with Standard Format beause they converge
!                   in such a way as only to approach and not to obtain the exact input value.  Nevertheless, the cababilty to
!                   test such radices exists here.
!
!                   Note that exception handling for log of a negative number was purposely not programmed and defaults to the
!                   True BASIC run time error of "LOG of number <= 0 [3004]".
!
!
! - - - - - Mainline processing - - - - -
!
!
!
CALL Input_radix
!
DO                                ! infinite loop; ESC key ends
   !
   CALL Input_pos_integer
   !
   CALL Set_initial_values
   !
   CALL Process_01
   !
LOOP
!
```

```
!
!
! - - - - - Subroutines below - - - -
!
!
SUB Input_radix
    !
    LET radix  = 0
    LET radix$ = ""
    !
    PRINT
    PRINT TAB(  5); "To stop his program, just close the window."
    PRINT
    PRINT
    PRINT TAB(  5); "To specify an integer radix other than the default of Phi, press ""Y"" or ""y"", otherwise press any key: ";
    !
    GET KEY radix_response
    PRINT
    !
    LET radix$ = LCase$( Chr$( radix_response))
    !
    IF radix$ = "y" THEN
       !
       PRINT
       PRINT TAB(  5);
       INPUT PROMPT "Please input a positive integer number radix: ": radix
       !
    END IF
    !
END SUB                          ! Input_radix
!
!
SUB Input_pos_integer
    !
    LET n      = 0
    !
    PRINT
    PRINT TAB(  5);
    INPUT PROMPT "Please input a positive integer: ": n
    PRINT
    !
END SUB                          ! Input_pos_integer
!
!
SUB Set_initial_values
    !
    ! LET n = 12                   ! ok 1-4; 5-6; 7, 8, 11; 12, 17; 18, ; 28, 29; 30; 65 ! not ok: 0,
    !
    LET inp_num2  = n
    LET mxd_num2  = inp_num2
```

```
    !
    IF radix = 0 THEN
       !
       LET phi2      = ( ( 5 ^ 0.5) + 1) / 2     !   <====  For any radix expression as the default, change the value of phi2 here.
       !
    ELSE                          ! radix <> 0
       !
       LET phi2 = radix
       !
    END IF
    !
    LET log_phi2  = log( phi2)
    LET max_log2  = log( mxd_num2) / log_phi2
    !
    LET max_pwr2  = INT( max_log2)
    LET max_pos2  = max_pwr2 + 1  ! to accommodate the zero exponent place for phi
    LET bin_len2  = max_pos2 * 2
    LET phi2$     = REPEAT$( "0", bin_len2)
    LET phi2_dec$ = ""
    LET phi2_end$ = ""
    LET min_pwr2  = ( - 1) * ( max_pwr2 + 1)     !  The range of powers is +( max_pwr2),..., +2, +1, 0, -1, -2,..., -( max_pwr2 + 1)
    !
    ! CALL Print_setup_variables
    !
    ! CALL Get_key_z
    !
END SUB                           ! Set_initial_values
!
!
SUB Process_01
    !
    LET phi_sum2 = 0
    !
    DO WHILE ( phi_sum2 < n)
       !
       LET cur_inp_num = inp_num2
       LET mxd_num2    = inp_num2      ! inp_num2 from previous run new calculation
       LET log_pwr2    = log( mxd_num2) / log_phi2    ! change 02.1
       !
       LET phi_log_pwr2  = INT( ROUND( log( mxd_num2) / log_phi2, 9))     ! testing all FLOOR, especially when negative
       LET phi_pwr2      = phi2 ^ phi_log_pwr2
       LET exp_pos2      = max_pos2 - phi_log_pwr2 ! pwr { 7, 6,..., 1, 0, -1,..., -7, -8} maps to pos { 1, 2,..., 7, 8, 9,..., 15, 16}
       !
       LET inp_num2                  = mxd_num2 - phi_pwr2     ! new inp_num2 calculation for next run
       LET nxt_inp_num2              = inp_num2
       LET phi2$[ exp_pos2: exp_pos2] = "1"      ! Set phi power bit on "1"
       LET phi_sum2                  = phi_sum2 + phi_pwr2     ! "+ ( phi2 ^ phi_log_pwr2)" == "phi_pwr2"
       !
       ! CALL Print_step_progress
       !
```

```
      ! CALL Get_key_z             ! Manually step through the loop
      !
    LOOP
    !
    LET phi2$     = phi2$[ 1: max_pos2] & "." & phi2$[ max_pos2 + 1: bin_len2]      ! Set decimal point for minimal "1" standard form
    LET phi2_dec$ = phi2$
    LET phi2_end$ = phi2_dec$
    LET bin_len2  = LEN( phi2_dec$)
    !
    DO WHILE phi2_dec$[ bin_len2: bin_len2] = "0"     ! Strip out trailing digit if "0"
      !
      LET phi2_end$ = phi2_dec$[ 1: bin_len2 - 1]
      LET bin_len2 = bin_len2 - 1     ! decrements the string length until non "0" is encountered
      !
    LOOP
    !
    CALL Print_final_result
    !
    ! CALL Get_key_z
    !
END SUB                           ! Process_01
!
!
SUB Print_setup_variables
    !
    PRINT
    PRINT TAB(  5); "Setup variables here:"
    PRINT
    PRINT "inp_num2 "; inp_num2
    PRINT "mxd_num2 "; mxd_num2
    PRINT "phi2     "; phi2
    PRINT "log_phi2 "; log_phi2
    PRINT "max_log2 "; max_log2
    PRINT "max_pos2 "; max_pos2
    PRINT "max_pwr2 "; max_pwr2
    PRINT "min_pwr2 "; min_pwr2
    PRINT "bin_len2 "; bin_len2
    PRINT "phi2$     "; phi2$; "  len2"; LEN( phi2$)
    PRINT
    PRINT TAB(  5); "Step wise processing below: "
    !
END SUB                           ! Print_setup_variables
!
!
SUB Print_step_progress
    !
    PRINT
    PRINT TAB(  0); "cur_inp_num2 "; cur_inp_num2; TAB( 25); "mxd_num2 "; mxd_num2; TAB( 50); "log_pwr2 "; log_pwr2
    PRINT TAB(  0); "phi_log_pwr2 "; phi_log_pwr2; TAB( 25); "phi_pwr2 "; phi_pwr2
    PRINT TAB(  0); "exp_pos2 "; exp_pos2; TAB( 50)
```

```
    !
    PRINT TAB(  0); "nxt_inp_num2 "; nxt_inp_num2; TAB( 25); "phi_sum2 ";  phi_sum2
    PRINT TAB(  0); "phi2$ wod "; phi2$
    !
END SUB                          ! Print_step_progress
!
!
SUB Get_key_z
    !
    GET KEY z
    !
    IF z = 27 THEN               ! ESC chracter
       !
       PRINT
       PRINT TAB(  5); "Program is stopped; this window closes automatically in 15-seconds."
       PRINT
       PAUSE 15
       STOP
       !
    END IF
    !
END SUB                          ! Get_key_z
!
!
SUB Print_final_result
    !
    PRINT
    ! PRINT TAB( 0); "phi2$ with    dec           "; phi2_dec$
    !
    ! PRINT TAB( 0); "phi2$ ends  in ""1"" or "".""   "; phi2_end$
    PRINT TAB( 0); "Result ends in ""1"" or "".""   "; phi2_end$
    !
END SUB                          ! Print_final result
!
!
END
```