```fortran
      PROGRAM ROOTS
      USE FMZM
      IMPLICIT NONE

!    Sample root-finding program.

!    FM_SECANT is a multiple precision root-finding routine.

!    The equation to be solved is  F(X,NF) = 0.
!    X  is the argument to the function.
!    NF is the function number in case roots to several functions are needed.

      CHARACTER(80) :: ST1
      TYPE (FM), SAVE     :: A1, A2, ROOT
      TYPE (FM), EXTERNAL :: F

!             Set the FM precision to 50 significant digits (plus a few "guard digits").

      CALL FM_SET(50)

!             Find a root of the first function,  X**2 - 3 = 0.
!             A1, A2 are two initial guesses for the root.

      A1 = 1
      A2 = 2

!             For this call no trace output will be done (KPRT = 0).
!             KU = 6 is used, so any error messages will go to the screen.

      WRITE (*,*) ' '
      WRITE (*,*) ' '
      WRITE (*,*) ' Case 1.  Call FM_SECANT to find a root between 1 and 2'
      WRITE (*,*) '          for f(x) = X**2 - 3.'
      WRITE (*,*) '          Use KPRT = 0, so no output will be done in the routine, then'
      WRITE (*,*) '          write the results from the main program.'

      CALL FM_SECANT(A1,A2,F,1,ROOT,0,6)

!             Write the result, using F35.30 format.

      CALL FM_FORM('F35.30',ROOT,ST1)
      WRITE (*   ,"(/'  A root for function 1 is ',A)") TRIM(ST1)


!             Find a root of the second function,  X*tan(X) - 1 = 0.  There are infinitely many
!             roots, and from the graph we decide to find the one between 6 and 7.

!             This time we ask for 50 digits of the root, and use FM_SECANT's built-in trace
!             (KPRT = 1) to print the final approximation to the root.  The output will appear on
!             more than one line, to allow for the possibility that precision could be hundreds or
!             thousands of digits, so the number might not fit on one line.

      WRITE (*,*) ' '
      WRITE (*,*) ' '
      WRITE (*,*) ' Case 2.  Find a root between 6 and 7 for  f(x) = x*tan(x) - 1.'
```

```
      WRITE (*,*) '           Use KPRT = 1, so FM_SECANT will print the result.'

      CALL FM_SECANT(TO_FM('6.0D0'),TO_FM('7.0D0'),F,2,ROOT,1,6)


!             Find a root of the third function,  gamma(x) - 10 = 0.  There is one root larger
!             than 1, and since gamma(5) is 24 this root is less than 5.

!             Get 50 digits of the root, and use FM_SECANT's built-in trace to print all
!             iterations (KPRT = 2) as well as the final approximation to the root.

      WRITE (*,*) ' '
      WRITE (*,*) ' '
      WRITE (*,*) ' Case 3.  Find a root between 1 and 5 for  f(x) = gamma(x) - 10.'
      WRITE (*,*) '           Use KPRT = 2, so FM_SECANT will print all iterations,'
      WRITE (*,*) '           as well as the final result.'

      CALL FM_SECANT(TO_FM(" 1.0 "),TO_FM(" 5.0 "),F,3,ROOT,2,6)


!             Find a root of the fourth function,  polygamma(0,x) = 0.
!             This root is the location of the one positive relative minimum for gamma(x),
!             since the derivative of gamma(x) is gamma(x)*polygamma(0,x).

!             Get 50 digits of the root, and use KPRT = 1 to print the root.

      WRITE (*,*) ' '
      WRITE (*,*) ' '
      WRITE (*,*) ' Case 4.  Find a root between 1 and 2 for  f(x) = polygamma(0,x).'
      WRITE (*,*) '           Use KPRT = 1, so FM_SECANT will print the result.'

      CALL FM_SECANT(TO_FM(" 1.0 "),TO_FM(" 2.0 "),F,4,ROOT,1,6)


!             Find a root of the fifth function,  cos(x) + 1 = 0.
!             This root has multiplicity 2 at x = pi.

!             Get 50 digits of the root, and use KPRT = 2 to print the iterations.

      WRITE (*,*) ' '
      WRITE (*,*) ' '
      WRITE (*,*) ' Case 5.  Find a root near 3.1 for  f(x) = cos(x) + 1.  (Double root)'
      WRITE (*,*) '           Use KPRT = 2, so FM_SECANT will print the iterations.'

      CALL FM_SECANT(TO_FM(" 3.1 "),TO_FM(" 3.2 "),F,5,ROOT,2,6)


!             Find a root of the sixth function,  cos(x) + 1 - 1.0D-40 = 0.
!             There are two different roots that agree to about 20 digits, so here
!             the convergence is slower.

!             Get 50 digits of the root, and use KPRT = 1 to print the root.

      WRITE (*,*) ' '
      WRITE (*,*) ' '
      WRITE (*,*) ' Case 6.  Find a root near 3.1 for  f(x) = cos(x) + 1 - 1.0E-40.'
```

```
      WRITE (*,*) '           There are two different roots that agree to about 20 digits,'
      WRITE (*,*) '           so here the convergence is slower.'
      WRITE (*,*) '           Use KPRT = 1, so FM_SECANT will print the result.'

      CALL FM_SECANT(TO_FM(" 3.1 "),TO_FM(" 3.2 "),F,6,ROOT,1,6)


!             Find a root of the seventh function,  sin(x) + ( x - pi ) = 0.
!             This root has multiplicity 3 at x = pi.

!             Get 50 digits of the root, and use KPRT = 2 to print the iterations.

      WRITE (*,*) ' '
      WRITE (*,*) ' '
      WRITE (*,*) ' Case 7.  Find a root near 3.1 for  f(x) = sin(x)**3.  (Triple root)'
      WRITE (*,*) '           Use KPRT = 2, so FM_SECANT will print the iterations.'

      CALL FM_SECANT(TO_FM(" 3.1 "),TO_FM(" 3.2 "),F,7,ROOT,2,6)

      WRITE (*,*) ' '

      END PROGRAM ROOTS

      FUNCTION F(X,NF)
      USE FMZM
      IMPLICIT NONE

!  X  is the argument to the function.
!  NF is the function number.

      INTEGER :: NF
      TYPE (FM) :: F, X

!             Functions create temporary multiple precision variables to hold the function values,
!             and also for argument values in cases where an argument might be A+B or TO_FM('1.7').
!             To avoid deleting these temporaries before we are finished using them, any function
!             that returns a multiple precision function value or has multiple precision arguments
!             must call FM_ENTER_USER_FUNCTION upon entry and FM_EXIT_USER_FUNCTION when returning.
!             The argument for both these routines is the function name, so the FM memory manager
!             will know when it is safe to delete these temporary variables.

      CALL FM_ENTER_USER_FUNCTION(F)
      IF      (NF == 1) THEN
          F = X*X - 3
      ELSE IF (NF == 2) THEN
          F = X*TAN(X) - 1
      ELSE IF (NF == 3) THEN
          F = GAMMA(X) - 10
      ELSE IF (NF == 4) THEN
          F = POLYGAMMA(0,X)
      ELSE IF (NF == 5) THEN
          F = COS(X) + 1
      ELSE IF (NF == 6) THEN
          F = COS(X) + (1 - TO_FM(' 1.0D-40 '))
      ELSE IF (NF == 7) THEN
          F = SIN(X)**3
```

```
ELSE
    F = 3*X - 2
ENDIF

CALL FM_EXIT_USER_FUNCTION(F)
END FUNCTION F
```