```fortran
      program test
      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.'
```

```fortran
      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 test

      function f(x, nf)     result (return_value)
      use fmzm
      implicit none

!  x  is the argument to the function.
!  nf is the function number.

      integer :: nf
      type (fm) :: return_value, x
      intent (in) :: x, nf

      if      (nf == 1) then
          return_value = x*x - 3
      else if (nf == 2) then
          return_value = x*tan(x) - 1
      else if (nf == 3) then
          return_value = gamma(x) - 10
      else if (nf == 4) then
          return_value = polygamma(0, x)
      else if (nf == 5) then
          return_value = cos(x) + 1
      else if (nf == 6) then
          return_value = cos(x) + (1 - to_fm(' 1.0d-40 '))
      else if (nf == 7) then
          return_value = sin(x)**3
      else
          return_value = 3*x - 2
      endif

      end function f
```