```
      program test
      use fmzm
      implicit none

!   This function was conjectured to have all of its roots on the unit circle:

!   f(x) = (b+1)/2*x^a - b*x^(a-1) + b*x^(a-2) - ... + b*x^2 - b*x + (b+1)/2
!          a is an even integer,  b is odd.
!          For this example function, a = 110,  b = 13.

!   This program confirms that in this case, all the roots are within 1e-50 of the unit circle.

!   For subroutine zm_roots, the equation to be solved is written  f(x, nf) = 0.
!   x  is the argument to the function.
!   nf is the function number in case roots to several functions are needed.

      integer :: j, kprt, n_found
      type (zm), external :: f
      type (fm) :: err
      type (zm), allocatable :: list_of_roots(:)

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

      call fm_set(50)
      call fm_setvar(" kswide = 130 ")

      allocate(list_of_roots(110), stat=j)
      if (j /= 0) then
          write (*, "(/' Error in program roots.  Unable to allocate arrays with n = ', i8/)") 110
          stop
      endif

!             Use kprt = 1, so zm_roots will print the results.

      kprt = 1

!             Find all roots of a 110th degree polynomial.  (nr = 110)

      write (*,*) ' '
      write (*,*) ' '
      write (*,*) ' Call zm_roots to find the roots of f(x) = ( 110th degree polynomial )'
      write (*,*) ' '

      call zm_roots(110, f, 1, n_found, list_of_roots, kprt, 6)

      write (*,*) ' '
      write (*,*) ' Sort these roots by argument and check how close they lie to the unit circle.'
      write (*,*) ' '

      call write_roots(6, n_found, list_of_roots)

!             Check that these roots all lie on the unit circle.

      err = 0
      do j = 1, n_found
         err = max(err, abs(1-abs(list_of_roots(j))))
      enddo
```

```
      write (*,*) ' '
      write (*, "(a, es16.7)") ' All roots lie on the unit circle within error ', to_dp(err)
      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 :: j, nf
      type (zm) :: return_value, x
      intent (in) :: x, nf

      if (nf == 1) then

!             (b+1)/2*x^a - b*x^(a-1) + b*x^(a-2) - ... + b*x^2 - b*x + (b+1)/2
!             a is an even integer,  b is odd.
!             For this example function, a = 110,  b = 13.
!             All the roots of this function lie on the unit circle.
!             This function has a simpler form, but is evaluated in this long form
!             to give a stronger check of roundoff control.

          return_value = (13+1)/2
          do j = 2, 110
             return_value = return_value*x + (-1)**(j-1)*13
          enddo
          return_value = return_value*x + (13+1)/2
      else if (nf == 2) then
          return_value = sin(x)
      else
          return_value = x**3 - 2
      endif

      end function f

      subroutine write_roots(ku, n_found, list_of_roots)
      use fmvals
      use fmzm
      implicit none

!  Write the list of roots in order of increasing argument on unit ku.

      integer :: i, j, jmin, j_printed(200), ku, kw_save, n_found
      type (fm) :: arg_root, size
      type (zm) :: list_of_roots(n_found)
      character(50) :: str
      double precision :: amin, args(200)
      intent (in) :: ku, n_found, list_of_roots

      kw_save = kw
      j_printed = 0
      do j = 1, n_found
         if (list_of_roots(j) == 0) then
```

```
        args(j) = 0
    else
        call zm_arg(list_of_roots(j), arg_root)
        args(j) = arg_root * 180 / acos(-1.0d0)
        if (args(j) < 0) args(j) = 360 + args(j)
    endif
enddo

do i = 1, n_found
    jmin = 1
    amin = 420
    do j = 1, n_found
       if (j_printed(j) == 0 .and. args(j) < amin) then
           jmin = j
           amin = args(j)
       endif
    enddo

    j_printed(jmin) = 1
    kw = ku
    size = abs(list_of_roots(jmin))
    call zm_print(list_of_roots(jmin))
    if (size < 1000) then
        write (ku, "(9x, 'Argument (degrees) = ', f15.10, 5x, 'Magnitude = ', f15.10)")  &
               amin, to_dp(size)
    else
        call fm_form(" es27.10 ", size, str)
        write (ku, "(9x, 'Argument (degrees) = ', f15.10, 5x, 'Magnitude = ', a)")  &
               amin, trim(str)
    endif
    write (ku,*) ' '
enddo

kw = kw_save

end subroutine write_roots
```