```
      program test
      use fmzm
      use fm_double_int
      use fm_quad_int
      use fm_quad_real

      implicit none

!  This program illustrates the use of the three modules for using FM, IM, or ZM variables
!  with double and quadruple length integer and real variables supplied by the compiler.

!  These would be needed for cases where the user's program uses all three precisions,
!  single, double, quad.  Then using a compiler switch to automatically make 64-bits the
!  default would not work.

!  The variables used to select the kind of integers and floating-point real and complex
!  numbers to be used, double_int, quad_int, and quad_fp, are defined in the three modules.


      integer ::                 j, j1, j2
      integer (double_int) ::  k, k1, k2
      integer (quad_int) ::    l, l1, l2

      real ::                    a1, a2
      real (kind(1.0d0)) ::    b1, b2
      real (quad_fp) ::        c1, c2

      complex ::                 r1, r2
      complex (kind(1.0d0)) :: s1, s2
      complex (quad_fp) ::     t1, t2

      type(fm) ::                f1, rel_error
      type(zm) ::                z1, z2

!            For a simple example calculation, sum several pieces of the harmonic series,
!            1/n + 1/(n+1) + ... + 1/(n+10**5).

!            n = 10**2  for single precision,
!            n = 10**13 for double precision,
!            n = 10**29 for quad precision.

!            In each case, use type(fm) to get a more accurate sum and compute the relative
!            error for the other sum.

      j1 = 10**2
      j2 = j1 + 10**5
      a1 = 0
      f1 = 0
      do j = j1, j2
         a2 = j
         a1 = a1 + 1/a2

         f1 = f1 + 1/to_fm(j)
      enddo
      rel_error = abs( (f1-a1)/f1 )
      write (*, "(/a, es15.7/)") ' Relative error for the single precision real sum = ',  &
                                 to_quad(rel_error)
```

```
      k1 = 10_double_int**13
      k2 = k1 + 10**5
      b1 = 0
      f1 = 0
      do k = k1, k2
         b2 = k
         b1 = b1 + 1/b2

         f1 = f1 + 1/to_fm(k)
      enddo
      rel_error = abs( (f1-b1)/f1 )
      write (*, "(/a, es15.7/)") ' Relative error for the double precision real sum = ',  &
                                 to_quad(rel_error)

      l1 = 10_quad_int**29
      l2 = l1 + 10**5
      c1 = 0
      f1 = 0
      do l = l1, l2
         c2 = l
         c1 = c1 + 1/c2

         f1 = f1 + 1/to_fm(l)
      enddo
      rel_error = abs( (f1-c1)/f1 )
      write (*, "(/a, es15.7/)") ' Relative error for the quad precision real sum   = ',  &
                                 to_quad(rel_error)

!             Do a similar sum using complex numbers.
!             1/sqrt(n+i) + 1/sqrt(n+1+i) + ... + 1/sqrt(n+10**4+i).

!             n = 10**2  for single precision,
!             n = 10**13 for double precision,
!             n = 10**29 for quad precision.

!             In each case, use type(zm) to get a more accurate sum and compute the relative
!             error for the other sum.

      j1 = 10**2
      j2 = j1 + 10**4
      r1 = 0
      z1 = 0
      z2 = to_zm(' i ')
      do j = j1, j2
         r2 = j + cmplx(0.0, 1.0)
         r1 = r1 + 1/sqrt(r2)

         z1 = z1 + 1/sqrt(j+z2)
      enddo
      rel_error = abs( (z1-r1)/z1 )
      write (*, "(/a, es15.7/)") ' Relative error for the single precision complex sum = ',  &
                                 to_quad(rel_error)

      k1 = 10_double_int**13
      k2 = k1 + 10**4
      s1 = 0
      z1 = 0
```

```
z2 = to_zm(' i ')
do k = k1, k2
   s2 = k + cmplx(0.0d0, 1.0d0, kind(1.0d0))
   s1 = s1 + 1/sqrt(s2)

   z1 = z1 + 1/sqrt(k+z2)
enddo
rel_error = abs( (z1-s1)/z1 )
write (*, "(/a, es15.7/)") ' Relative error for the double precision complex sum = ',  &
                           to_quad(rel_error)

l1 = 10_quad_int**29
l2 = l1 + 10**4
t1 = 0
z1 = 0
z2 = to_zm(' i ')
do l = l1, l2
   t2 = l + cmplx(0.0_quad_fp, 1.0_quad_fp, quad_fp)
   t1 = t1 + 1/sqrt(t2)

   z1 = z1 + 1/sqrt(l+z2)
enddo
rel_error = abs( (z1-t1)/z1 )
write (*, "(/a, es15.7/)") ' Relative error for the quad precision complex sum   = ',  &
                           to_quad(rel_error)

end program test
```