

1. Here in text is a cartesian representation of a 3-D filter. Draw a circle around the starting end of the helix and an arrow to denote direction of increasing lag. Call this direction the 1-axis. What are the three numerical values of the three components of the `ra` vector? What are the three numerical values of the three components of the `center` vector?

```

0.000  0.000  0.000  0.000  0.000
0.000  0.000  1.000  2.000  2.000
2.000  2.000  2.000  2.000  2.000
-----
2.000  2.000  2.000  2.000  2.000
2.000  2.000  2.000  2.000  2.000
2.000  2.000  2.000  2.000  2.000

```

2. What are the inputs to `createhelixrod` ?

```

module createhelixrod {
  use helix + cartesian
  contains
  function createhelix( rd, center, gap, ra) result( aa) {
    type( filter)           :: aa      # needed by helicon.
    integer, dimension(:), intent(in) :: rd, ra  # data and filter axes
    integer, dimension(:), intent(in) :: center  # normally (ra/2,ra/2,...,1)
    integer, dimension(:), intent(in) :: gap     # normally ( 0,  0,  0,...,0)
    integer, dimension( size( rd))     :: ii     # cartesian indexes
    integer                  :: ra123, ia, ndim, rh, lag0a,lag0d
    integer, dimension(:), allocatable:: lag
    rh= 0;  ra123 = product( ra);  ndim = size( rd)
    allocate( lag( ra123 ) )      # filter cube size
    call cart2line( ra, center, lag0a) # lag0a = index pointing to the "1.0"
    do ia = 1+lag0a, ra123 {      # ia is fortran linear index.
      call line2cart( ra, ia, ii)  # ii(ia) is fortran array indices.
      if( any( ii <= gap)) next   # ignore some locations
      rh = rh + 1                 # got another live one.
      call cart2line( rd, ii, lag(rh)) # get its fortran linear index
    }
    call cart2line( rd, center, lag0d) # lag0d is center shift for rd_cube
    call allocatehelix( aa, rh)      # rh becomes size of filter on helix.
    aa%lag = lag(1:rh) - lag0d;     # lag = fortran_linear_index - center
    aa%flt = 0.0;                   deallocate( lag)
  }
}

```

3. In the code `createhelixrod` circle the three lines which create the output (directly, not indirectly).