GP211. Closed book. 7 minutes. January 7, 2014. Your name _____

Today is your opportunity to predict ahead of the drill bit. You measure a real scalar variable at each well in an oil field. You also estimate the same variable from much denser seismic coverage. You observe a systematic discrepancy between the wells and the seismic data. You theorize that the discrepancy arises from seismic anisotropy, but you cannot convince your struggling data processing contractor to install your anisotropy processing and rerun it free of charge.

You prepare a map $\mathbf{m}$ of the discrepancy which you know only at the wells – a missing data problem. You find the code required in Claerbout's book, but he minimizes $\nabla^2\mathbf{m} \approx \mathbf{0}$ and you want to minimize $\nabla\mathbf{m} \approx \mathbf{0}$. You see two modules below. Read the laplacian module to learn about the gradient module. Then MAKE YOUR CHANGES IN THE BIN-FILLING MODULE. (IMPORTANT DETAIL: Recall sophomore mathematics that the output of ∇^2 is a scalar whereas the output of ∇ is a vector so in 2-D you need to have twice as much memory for output.) HINT: Don't build the gradient operator. Use the same one the Laplacian operator is using.

```
module laplac2 {                                    # Laplacian operator in 2-D
use igrad2
real, dimension (m1*m2*2), allocatable :: tmp
#%_init    (m1, m2)
    integer m1, m2
    call igrad2_init (m1, m2)
#%_lop (x, y)
    integer stat1, stat2
    if( adj) {
        stat1 = igrad2_lop (.false., .false., y, tmp)   # tmp = grad y
        stat2 = igrad2_lop ( .true.,    add , x, tmp)   # x = x + grad' tmp
    } else   {
        stat1 = igrad2_lop (.false., .false., x, tmp)   # tmp = grad x
        stat2 = igrad2_lop ( .true.,    add , y, tmp)   # y = y + grad' tmp
    }
}
module lapfill {   # fill empty 2-D bins by minimum output of Laplacian operator
  use laplac2
  use cgstep_mod
  use solver_mod
contains
  subroutine lapfill2( niter, m1, m2, xx, mfixed) {
    integer,                    intent (in)     :: niter  # iterations
    integer,                    intent (in)     :: m1,m2  # data size
    logical, dimension (m1*m2), intent (in)     :: mfixed # mask for known
    real,    dimension (m1*m2), intent (in out) :: xx     # model

    real,    dimension (m1*m2)                  :: zero   # laplacian output

    call laplac2_init ( m1,m2);            zero = 0.      # initialize

    call solver ( laplac2_lop, cgstep, xx,  zero, niter,
                               x0 = xx,  known = mfixed)
    call laplac2_close ()                                 # garbage collection
    call cgstep_close ()                                  # garbage collection
  }
}
```

Did you need to change the size of the mask `mfixed` ?