CPS 5320 (March 2017) Qualifier

1. To derive the minimal surface equation, suppose $u(x, y)$ is the surface with $u = g(x, y)$ on the boundary $\partial\Omega$ of Ω which minimizes the surface area $SA(u) \equiv \int\int_\Omega \sqrt{1 + u_x^2 + u_y^2}\ dA$. Then if $e(x, y)$ is any smooth function with $e = 0$ on the boundary, $SA(u + \alpha e) \geq SA(u)$ for any α. From this we conclude that $f(\alpha) \equiv SA(u + \alpha e)$ has a minimum at $\alpha = 0$, and thus $f'(0) = 0$. Write out the equation $f'(0) = 0$ and then derive a partial differential equation for u.

   Hint: you can use the multidimensional integration by parts formula:

$$\iint_\Omega \nabla u \bullet \mathbf{v} = \int_{\partial\Omega} u\mathbf{v} \bullet n - \iint_\Omega u\nabla \bullet \mathbf{v}$$

   where u is a scalar function, $\mathbf{v}$ is a vector function.

2. Solve this minimal surface equation using PDE2D, in the region $1 < r < 2, 0 < \theta < \pi/2$, with $U = (r-1)(2-r)$ on the boundary, where r, θ are polar coordinates. Output the integral of U over the region (should be about 0.101).

3. a. Modify the ?????s in the following code so that each processor initializes the elements of N-vectors X and Y only on its processors, and calculates the dot product X and Y in parallel, when the elements of X and Y are distributed cyclically over the NPES processors, that is, in the order
0,1,2,...,NPES-1,0,1,2...,NPES-1,0,1,2,...,NPES-1,...

```
      PARAMETER (N=12000000)
      DOUBLE PRECISION X(N),Y(N),SUM,SUMI
      INCLUDE 'mpif.h'
C                       INITIALIZE MPI
      CALL MPI_INIT (IERR)
C                       NPES = NUMBER OF PROCESSORS
      CALL MPI_COMM_SIZE (MPI_COMM_WORLD,NPES,IERR)
C                       ITASK = MY PROCESSOR NUMBER
      CALL MPI_COMM_RANK (MPI_COMM_WORLD,ITASK,IERR)
      NB = N/NPES
      DO I=1,N
         IF (?????.EQ.ITASK) THEN
            X(I) = I
            Y(I) = I+2
         ENDIF
      ENDDO
C
      SUMI = 0.0
      DO I=?????
         SUMI = SUMI + X(I)*Y(I)
      ENDDO
      CALL MPI_ALLREDUCE(SUMI,SUM,1,MPI_DOUBLE_PRECISION,
     &                   MPI_SUM,MPI_COMM_WORLD,IERR)
      PRINT *, SUM
      CALL MPI_FINALIZE(IERR)
      STOP
      END
```

b. Same question, but now assume the elements of X and Y are distributed over the processors by blocks, that is, in the order (you may assume N is divisible by NPES):
0,0,...0,1,1,...,1,2,2,...2,...

```
      PARAMETER (N=12000000)
      DOUBLE PRECISION X(N),Y(N),SUM,SUMI
      INCLUDE 'mpif.h'
C                       INITIALIZE MPI
      CALL MPI_INIT (IERR)
C                       NPES = NUMBER OF PROCESSORS
      CALL MPI_COMM_SIZE (MPI_COMM_WORLD,NPES,IERR)
C                       ITASK = MY PROCESSOR NUMBER
      CALL MPI_COMM_RANK (MPI_COMM_WORLD,ITASK,IERR)
      NB = N/NPES
      DO I=1,N
         IF (?????.EQ.ITASK) THEN
            X(I) = I
            Y(I) = I+2
         ENDIF
      ENDDO
C
      SUMI = 0.0
      DO I=?????
         SUMI = SUMI + X(I)*Y(I)
      ENDDO
      CALL MPI_ALLREDUCE(SUMI,SUM,1,MPI_DOUBLE_PRECISION,
     &                   MPI_SUM,MPI_COMM_WORLD,IERR)
      PRINT *, SUM
      CALL MPI_FINALIZE(IERR)
      STOP
      END
```

c. Would you expect either code (a) or (b) to be significantly faster than the other? Explain.

4. What will be the output from the program below, if it is run on NPES=3 processors?

```
      PARAMETER (N=4)
      DOUBLE PRECISION A(N),B(N)
      INCLUDE 'mpif.h'
C                       INITIALIZE MPI
```

```
      CALL MPI_INIT (IERR)
C                     NPES = NUMBER OF PROCESSORS
      CALL MPI_COMM_SIZE (MPI_COMM_WORLD,NPES,IERR)
C                     ITASK = MY PROCESSOR NUMBER
      CALL MPI_COMM_RANK (MPI_COMM_WORLD,ITASK,IERR)
      DO I=1,N
         A(I) = 10*ITASK + I
         B(I) = A(I)
      ENDDO
C
      iroot = 2
      CALL MPI_BCAST(A,N,MPI_DOUBLE_PRECISION,iroot,
     &                MPI_COMM_WORLD,IERR)
      CALL MPI_REDUCE(A,B,N,MPI_DOUBLE_PRECISION,
     &                MPI_SUM,iroot,MPI_COMM_WORLD,IERR)
      PRINT *, ITASK,B
      CALL MPI_FINALIZE(IERR)
      STOP
      END
```