Old Questions (with answers)

Name: _____

Part I. *Multiple choice.* One point each.

C 1. Which of the following is *not* a Fortran keyword?

a. **if** b. **open** c. **output** d. **write** e. **do** f. **exit**

F 2. How many times will the code inside the following `DO`-loop be executed? (Assume all variables are properly declared `INTEGER` variables, and `c` has already been set to some value.)

```
do j=c,c+4
   ⋮
end do
```

a. 0 b. 1 c. 2 d. 3 e. 4 f. 5

E 3. Which of the following is *not* a valid Fortran statement? (They are unrelated individual statements, not a piece of a program.)

a) `integer :: j, k, day`
b) `real :: height, pressure, temperature`
c) `write (unit=*,fmt=*) 'Enter state population'`
d) `newpop = (1.0 + rate) * pop`
e) `18.0 = t0 - (period / rate) ** 2`

A 4. What statement about the following Fortran statement is *false*?

```
write (unit=*,fmt='("Ozone level is",F5.0," in ",I5)') dobsons, year
```

a) `dobsons` and `year` must both be of type `REAL`.
b) `dobsons` will be written out to the nearest whole number value.
c) The statement will write information to the terminal screen.
d) The values of `dobsons` and `year` will be unchanged by this statement.
e) `dobsons` and `year` should have been given values (for example, by a `READ` statement or an arithmetic statement) before this statement is executed.

A 5. The following block of code contains *one* syntax error. Indicate which line is incorrect. (Assume variables have been properly declared and initialized.)

a) `if (x > 0.01)`
b) `  do k=2,n,2`
c) `    x = x * z(k)`
d) `  end do`
e) `end if`

D 6. `r` has the value 3 and `s` has the value −4. Which one of the following logical expressions is *true*?

a) `s > r`
b) `abs(s) <= abs(r)`
c) `abs(r + s) > abs(r)`
d) `abs(r) > s`
e) `abs(s) < r`

D 7. Which of the following Fortran intrinsic functions can only operate with an array argument?

a. ***max***	c. ***log***	e. ***abs***
b. ***min***	d. ***sum***	f. ***nint***

C 8. Which statement about the following `IF` structure is true?

```
if (a < 0.0) THEN
    a = b
else if (a < 1.0) THEN
    a = c
else
    a = b * c
end if
```

a) Nothing will happen to `a` if `a` is greater than one.
b) If `a` is greater than zero, it will always be set equal to the value of `c`.
c) If `a` is greater than one, then its value will be changed to `b` × `c`.
d) Two logical expressions will be evaluated every time this **if** block is encountered.

A 9. Pick the Fortran statement that does not necessarily define or mention an array name. (These statements do not necessarily fit together as a piece of a program.)

a) **if** (***max***`(i,j)` > ***sum***`(k))` **THEN**
b) **character**`(30) :: d(10)`
c) **integer** `:: a, b, c(20)`
d) **write** `(8,'(A)') h(j:k)`
e) `g =` ***maxval*** `( p )`

E 10. After the following declaration, how many real numbers are being stored?

```
real, dimension(10) :: u, v
```

a. 2	c. 11	e. 20
b. 10	d. 12	f. 21

D 11. For a subroutine contained in a module, which of the following *will* be *checked by the compiler* when comparing actual arguments to dummy arguments?

a) The *shape* of the actual arguments must match the shape of the dummy arguments.
b) The *order* of the actual arguments must match the order of the dummy arguments, even with keyword arguments. declarations and corresponding actual arguments.
c) The *rank* of the actual arguments must match the type of the dummy arguments. (Rank means scalar, one-dimensional array, two-dimensional array, etc.)
d) The *size* of arrays must match between dummy arguments and actual arguments.
e) The *name* of the dummy arguments must be the same as the actual arguments.

Consider this short program for the following three multiple choice questions.

```
program GRAVITY
implicit none
real ::  g, z
integer ::  j
real, parameter ::  g0 = 9.8, a=6370.0
write (unit=*,fmt='("Height Gravity")')
do j=0,100,10
    z = real(j)
    g = g0 / (1.0 + z / a) ** 2
    write (unit=*,fmt='(I5,4X,F7.5)') z, g
end do
stop
end program GRAVITY
```

C 12. Which of these equations is being calculated for various values of z?

a. $g = g_0 \left[1 + \left(\frac{z}{a} \right)^{-2} \right]$

b. $g = g_0 \left[1 + \left(\frac{z}{a} \right)^{2} \right]$

c. $g = g_0 \left(1 + \frac{z}{a} \right)^{-2}$

d. $g = g_0 \left(1 + \frac{z}{a} \right)^{2}$

e. $g = g_0 \left(1 + \frac{z^2}{a^2} \right)$

f. $g = g_0 \left(1 + \frac{a^2}{z^2} \right)$

D 13. How many *symbolic names* are defined in this program?

a. 3 b. 4 c. 5 d. 6 e. 7 f. 8

C 14. How many passes will be made through the **do** loop?

a. 0, b. 10, c. 11, d. 100, e. 101, f. 110

C 15. Which of the following is *not* a legal Fortran symbolic name?

a. `My_cat`
b. `h8tz`
c. `__DOGS`
d. `Every_Time`
e. `It5me11s`
f. `them`

A 16. You have a data file 425 lines long with 8 numbers on each line. To work with the data, you set up an array in which to store the numbers with the declaration

```
integer, parameter:: nl=425, ns=8
real, dimension(nl,ns) :: b
integer :: j, k
```

Which of the following code blocks will successfully read the file into `b`? (Assume that the data file has been opened to unit 20.)

a.
```
do j=1,nl
  read (unit=20,fmt=*) b(j,1:ns)
end do
```

b.
```
do j=1,ns
  read (unit=20,fmt=*) b(:,j)
end do
```

c.
```
do j=1,nl
  do j=1,ns
    read (unit=20,fmt=*) b(j,k)
  end do
end do
```

d.
```
do j=1,ns
  read (unit=20,fmt=*) b(1:nl,j)
end do
```

e.
```
do j=1,nl
  read (unit=20,fmt=*) b(1:ns,j)
end do
```

f.
```
do j=1,ns
  do j=1,nl
    read (unit=20,fmt=*) b(k,j)
  end do
end do
```

E 17. How many numbers will be printed by the following statements?

```
real :: a, b(2), c(20)
   ⋮
write (unit=*,fmt=*) a, b, c(1:10)
```

a. 3 b. 4 c. 10 d. 12 e. 13 f. 23

E 18. What is the maximum number of dimensions (maximum rank) of a Fortran array?

a. 3 b. 4 c. 5 d. 6 e. 7 f. 8

E 19. Which Fortran statement is a correct analogue for the equation

$$y = ae^{x^2/2}$$

a. `y = a * exp * ( 0.5 * x * x )`
b. `y = a * exp ( 0.5 / x ** 2 )`
c. `y = a * exp ** ( 0.5 * x * x )`
d. `y = a * exp ( 2.0 / x ** 2 )`
e. `y = a * exp ( 0.5 * x ** 2 )`
f. `y = a * exp * ( x ** 2 / 2.0 )`

B 20. What will be printed to the screen by the following Fortran statements? (␣ denotes a blank space.)

```
character(len=6) ::  fm='wool␣␣', fn = 'lamb␣␣'
character(len=13) ::  name
name = trim(fm) // '.'  // fn
write (unit=*,fmt=*) trim(name)
```

a. `wool␣␣.la`
b. `wool.lamb`
c. `wool␣␣.lamb`
d. `wool␣␣.lamb␣␣`
e. `wool.lamb␣␣␣␣`
f. `wool`

C 21. Which statement about the following **do** loop is *false*?

```
do j=k,m
   ⋮
end do
```

a) The **do**-variable `j` must not be changed by arithmetic inside the loop.
b) The **do**-variable `j` must be **integer**.
c) Only one pass through the loop will be taken if `k` is greater than `m`.
d) The "stride" will be +1 by default, since no stride variable is specified.

D 22. What is the value of the integer variable `k` in the following Fortran arithmetic expression?

```
k = 11 - 3 ** 2 / 4 * 2
```

a. 1 b. 5 c. 6 d. 7 e. 8 f. 9

E 23. After the following declaration, how many real numbers are being stored?

```
real, dimension(8) ::  u, v, c
```

a. 3 b. 8 c. 10 d. 16 e. 24 f. 27

D 24. Considering the following piece of a Fortran program.

```
program FINAL2
    use SOME_LIB, only : SOMETHING
    implicit none
    integer, parameter :: n=10
    real :: a, b(2), c(n), d(1:n,3)
        ⋮
    call SOMETHING ( w=a, x=b, y=d(1:n,1), z=d(1:n,3) )
    write (6,*) a, b, d(1,1:3)
    stop
end program FINAL2
```

Which of the following blocks could *not* be part of a legal beginning for subroutine SOMETHING?

a)
```
subroutine SOMETHING ( w, x, y, z )
real, intent(inout), dimension(:) :: x, y, z
real, intent(out) :: w
```

b)
```
subroutine SOMETHING ( w, x, y, z )
real, intent(out), dimension(:) :: x, y, z
real, intent(inout) :: w
```

c)
```
subroutine SOMETHING ( w, x, y, z, t )
integer, intent(in), dimension(:,:), optional:: t
real, intent(out) :: w, x(:), y(:), z(:)
```

d)
```
subroutine SOMETHING ( w, x, y, z )
real, intent(out) :: w
real, intent(inout) :: x(:), y(:,1), z(:,3)
```

e)
```
subroutine SOMETHING ( w, x, y, z )
real, intent(inout) :: w, x(:), y(:), z(:)
```

D 25. Which of the following Fortran keywords is *not* associated with defining and using modules and subroutines, or with declaring variables inside these scoping units.

a. **call** b. **use** c. **return** d. **open** e. **intent** f. **optional**

B 26. Which of the following is *not* a way to declare a character array consisting of 10 character strings, each of which is 20 characters long?

a) `character(20), dimension(10) :: a`
b) `character(10) :: a(20)`
c) `character(len=20), dimension(10) :: a`
d) `character(20) :: a(10)`

A 27. Which of the following intrinsic functions does *not* use characters or character-strings as its argument(s).

a. ***minloc*** c. ***iachar*** e. ***len_trim***
b. ***index*** d. ***trim*** f. ***scan***

Part III. *Programming.* Each problem is worth 10 points.

28. You have a file called `bird.census` which contains $n = 1981$ lines, each representing information from a different Audubon Society Christmas Bird Count region. Each line contains three numbers: the number of person-hours spent looking for birds H, the number of different species seen S, and the total number of birds seen B. Write a Fortran program which reads this file and
    - calculates the average *rates* at which birds as species were seen, as R_b = birds seen per person-hour and R_s = species seen per person-hour, averaged over the entire set
    $$R_\mathrm{b} = \frac{1}{n}\sum_{j=1}^{n}\frac{B_j}{H_j} \qquad\qquad R_\mathrm{s} = \frac{1}{n}\sum_{j=1}^{n}\frac{S_j}{H_j}$$
    - writes a new file called `bird.rates` that contains one line for each region, and each line contains two numbers. Those numbers are the *ratios* of the bird-finding and species-finding rates for each region to the the overall averages of those rates
    $$\dot{b}_j = \frac{B_j}{H_j \cdot R_\mathrm{b}} \qquad\qquad \dot{s}_j = \frac{S_j}{H_j \cdot R_\mathrm{s}}$$

Do not forget to use Fortran variable names and Fortran **real** arithmetic.

```
program CHIRP
! No array arithmetic.
    implicit none
    integer, parameter ::  n=1981
    real, dimension(n) ::  h, s, b
    real ::  rb, rs, ds, db
    integer ::  j
    open (unit=10,file='bird.census')
    open (unit=11,file='bird.rates')
    rs = 0.0
    rb = 0.0
    do j=1,n
        read (unit=10,fmt=*) h(j), s(j), b(j)
        rs = rs + s(j) / h(j)
        rb = rb + b(j) / h(j)
    end do
    rs = rs / n
    rb = rb / n
    do j=1,n
    ds = s(j) / (h(j) * rs)
    db = b(j) / (h(j) * rb)
        write (unit=11,fmt=*) ds, db
    end do
    stop
end program CHIRP
```

```
program CHIRP
! Use array arithmetic, output calculated in I/O list
    implicit none
    integer, parameter ::  n=1981
    real, dimension(n) ::  h, s, b
    real ::  rb, rs
    integer ::  j
    open (unit=10,file='bird.census')
    open (unit=11,file='bird.rates')
    do j=1,n
        read (unit=10,fmt=*) h(j), s(j), b(j)
    end do
    rs = sum ( s / h) / n
    rb = sum ( b / h) / n
    do j=1,n
        write (unit=11,fmt=*) s(j) / (h(j) * rs), b(j) / (h(j) * rb)
    end do
    stop
end program CHIRP
```

29. You have a file called `bridger.basin` which contains 135 lines, each representing a different small land-management unit of a region. Each line contains three numbers: the number of sheep grazing on that land (N), the area in hectares of pasture land in the subregion (A), and a carrying capacity (C) that indicates how many sheep-per-hectare that land can sustain. Write a Fortran program which
    - reads this file,
    - calculates a total carrying capacity for the region. $T = \sum_{j=1}^{135} C_j \cdot A_j$.
    - writes a new file called `basin2` that contains one line for each subregion, and each line contains two numbers: the *percentage* of sheep to total carrying capacity of the region, $P_j = 100 \times N_j \div T$ and the difference between how many sheep are being grazed on the unit and its carrying capacity, $D_j = N_j - C_j \cdot A_j$.

    Don't forget to use Fortran **real** arithmetic where needed. (10 points)

```
program BAA
    implicit none
    integer, parameter :: n=135
    real, dimension(n) :: ns, a, c, p, d
    real :: t
    integer :: j

    open (unit=10,file='bridger.basin')
    do j=1,n
        read (unit=10,fmt*) n(sj), a(j), c(j)
    end do

    t = dot_product ( c, a )
    p = 100.0 * ns / t
    d = ns - c * a

    open (unit=11,file='basin')
    do j=1,n
        write (unit=11,fmt=*) p(j), d(j)
    end do
    stop
end program BAA
```

```
program BAA
    implicit none
    integer, parameter :: n=135
    real, dimension(n) :: ns, a, c
    real :: t, p, d
    integer :: j

    open (unit=10,file='bridger.basin')
    t = 0.0
    do j=1,n
        read (unit=10,FMT=*) ns(j), a(j), c(j)
        t = t + a(j) * c(j)
    end do

    open (unit=11,file='basin')
    do j=1,n
        p = 100.0 * ns(j) / t
        d = ns(j) - c(j) * a(j)
        write (unit=11,fmt=*) p, d
    end do
    stop
end program BAA
```

30. You have a data file named `shellpot.raw` which contains 37 lines of data, arranged in 9 columns. Each line represents a different sampling location, and each column represents a particle-size class (a range of sediment particle diameters). Write a program that (1) reads the data, (2) calculates the total mass of sediment in each particle size class over all 37 locations, (3) divides each individual sample mass by the total mass of its particle size [i.e., divide each individual sample by the size class total calculated in (2)], and (4) writes a new file called `shellpot.props` that contains the proportions you have calculated, organized in 37 lines of 9 columns similar to the file you read. (12 points)

```
program RiverBottom
    implicit none
    integer, parameter :: nsize=9, nloc=37
    real, dimension(nloc,nsize) :: mass
    real, dimension(nsize) :: tmass
    integer :: j

    open (unit=1,file='shellpot.raw')
    open (unit=2,file='shellpot.props')
    do j=1,nloc
        read(unit=1,fmt=*,action='read') mass(j,1:nsize)
    end do

    tmass = sum ( mass, dim=1 )
    do j=1,nloc
        write(unit=2,fmt=*) mass(j,1:nsize) / tmass(1:nsize)
    end do

    stop
end program RiverBottom
```

```
program RiverBottom
    implicit none
    integer, parameter :: nsize=9, nloc=37
    real, dimension(nloc,nsize) :: mass
    real, dimension(nsize) :: tmass
    integer :: j, k

    open (unit=1,file='shellpot.raw')
    open (unit=2,file='shellpot.props')
    do j=1,ncols
        read(unit=1,fmt=*,action='read') mass(j,1:nsize)
    end do

    do k=1,nsize
        tmass(k) = 0.0
        do j=1,nloc
            tmass(k) = tmass(k) + mass(j,k)
        end do
    end do

    do j=1,nloc
        do k=1,nsize
            mass(j,k) = mass(j,k) / tmass(k)
        end do
        write(unit=2,fmt=*) mass(j,1:nsize)
    end do

    stop
end program RiverBottom
```