

Grade: _____/100
130

- 25

100 Format (A8, A14)

J o h n					, H i B a b .																	
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23

[illegible]

- ```
OPEN (UNIT=22, FILE="iAData.dat", STATUS='OLD', &
ACTION='READ', IOSTAT=err1)
```

```
OPENC UNIT=44, FILE="OutData.dat", STATUS='UNKNOWN'
ACTION='WRITE', &STAT=err2)
```

```
IF (err1/=0 OR err2/=0) THEN
 WRITE(*,*) "error"
 STOP
```

END II

3. (15 pts) You want to create your own Fortran SUBROUTINE definition that will display to the screen two real values in the Cartesian coordinate format. For example, if  $x$  and  $y$  are real variables, equal to 3.1 and 4.2, the SUBROUTINE would display to the screen:

(3.1, 4.2)

Write a complete SUBROUTINE definition called *displayCart* that performs this operation.

```
SUBROUTINE displayCart(x, y)
 REAL, INTENT(IN) :: x, y
 WRITE(*,*) "(, ", x, ", ", y, ")"
END SUBROUTINE
```

4. (10 pts) Declare arrays and initialize values.  
 declare a real array of size 30 called *realNums*  
 declare a integer array of size 30 called *intNums*  
 declare a logical array of size 30 called *boolVals*

Initialize all values in *realNums* to 0.0, all values in *intNums* to 0 and all values in *boolVals* to .FALSE..

```
INTEGER, PARAMETER :: aSize = 30
REAL, DIMENSION(aSize) :: realNums = 0.0
INTEGER, DIMENSION(aSize) :: intNums = 0
LOGICAL, DIMENSION(aSize) :: boolVals = .FALSE.
```

5. (10 pts) Write code to transfer all the values in a real array of size 50 called array1 to the corresponding locations of a real array of size 50 called array2. Use a do loop and declare all variables used.

```

REAL, DIMENSION(50):: array1, array2
INTEGER:: i
DO i = 1, 50
 array2(i) = array1(i)
END DO

```

6. Write a piece of code that declares an array of characters called **Names**. There should be 10 **Names** that are 15 characters long each (write code for this). Write code that you would use to get a list of 10 names from the user. (10 points)

```

CHARACTER(LEN=15), DIMENSION(10):: Names
WRITE(*,*) 'Please enter 10 Names'
DO i = 1, 10
 READ(*,*) Names(i)
END DO

```

7. (10 pts) Write code to declare a two dimensional array of type INTEGER that has 10 rows and 10 columns. Assign values to the array using counting loops, where the values look like a multiplication table. i.e.

|   |   |    |    |    |    |    |    |    |    |
|---|---|----|----|----|----|----|----|----|----|
| 1 | 2 | 3  | 4  | 5  | 6  | 7  | 8  | 9  | 10 |
| 2 | 4 | 6  | 8  | 10 | 12 | 14 | 16 | 18 | 20 |
| 3 | 6 | 9  | 12 | 15 | 18 | 21 | 24 | 27 | 30 |
| 4 | 8 | 12 | 16 | 20 | 24 | 28 | 32 | 36 | 40 |

etc.

```

INTEGER, DIMENSION(10,10):: ar
INTEGER:: i, j
DO i = 1, 10
 DO j = 1, 10
 ar(i,j) = i * j
 END DO
END DO

```

8. (30 pts)  $x!$  is defined as  $x$  factorial and is computed by calculating the product of all the values from  $x$  down to 1.

Some examples,  $5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$ ;  $2! = 2$  and, by definition of this function  $0! = 1$ .

Write a complete function name Factorial that implements this functionality. Your function should accept a single integer value and produce the resulting factorial answer. If any error occurs ( the function is sent a negative value for input ), your function should return -1.

```
INTEGER FUNCTION Factorial (n)
 IMPLICIT NONE
 INTEGER, INTENT(IN) :: n
 INTEGER :: i
 Factorial = 1
 DO
 IF (n >= 0) THEN
 DO i = 1, n
 Factorial = Factorial * i
 ENDDO
 ELSE
 Factorial = -1
 ENDIF
END FUNCTION
```

9. (20 pts) Write a Fortran SUBROUTINE called intSortArray. The subroutine is sent, on the parameter list, an array of INTEGERS called arr and an INTEGER called count. The code in the subroutine should sort the values in the array from low to high.

The layout / header to the Subroutine will look like:

```
SUBROUTINE intSortArray (arr, count)
```

! Your code goes here

```
IMPLICIT NONE
INTEGER, DIMENSION(count), INTENT(INOUT):: arr
INTEGER, INTENT(IN):: count
INTEGER:: i, j, tmp
DO i = 1, count-1
 DO j = 1, count-1
 IF (arr(j) > arr(j+1)) THEN
 tmp = arr(j)
 arr(j) = arr(j+1)
 arr(j+1) = tmp
 ENDIF
 END DO
END DO
```

```
END SUBROUTINE
```

