Eg 5.1

I/ See formulation in question Set 2 Q 5.

II/ Read a line of the data file
Classify the student
Print result
Repeat

III/

```
PROGRAM
! Description
! Declare variables
! Print header:
! Loop until end-of-file.
10  Read line of file (SID, HOURS)
    ! Classify student
    IF (0<HOURS.AND. HOURS<=30) &
          CLASS = 'freshman'
    ! IF (0< HOURS <=30) in FORTRAN
    ! fails.
                etc
    ! Print line of output:
    PRINT SID, HOURS, CLASS
    ! End of loop
    GO TO 10
    END
```

IF block

IV

```
      PROGRAM student
      ! Description: classify students by hours.
      ! Declare variables:
      INTEGER :: SID, HOURS
      CHARACTER (len=9) :: CLASS
      ! Print header:
      WRITE(*,*)  ! Prints blank line.
      WRITE(*,*)
      WRITE(*,*) '      REGISTRATION REPORT'
      WRITE(*,*) 'STUDENT  HOURS CLASSIFICATION'
      ! Open file:
      OPEN(1, FILE = 'student.dat')
10    READ(1,*, END=20) SID, HOURS
      ! Classify student:
      IF (0<HOURS.AND. HOURS<=30) THEN
        CLASS = 'Freshman'
      ELSEIF(30<HOURS.AND.HOURS<=60)THEN
        CLASS = 'Sophomore'
      ELSEIF (60<HOURS.AND. HOURS<= 90)THEN
        CLASS = 'Junior'
      ELSEIF (90<HOURS) THEN
        CLASS = 'Senior'
```

```
      ELSE
        CLASS = 'Unknown'
      ENDIF
      WRITE (*,30) SID, HOURS, CLASS
   30 FORMAT (T5, I4, T20, I3, T40, A)
      GO TO 10
   20 STOP          ! Replace by other code for Part III.
      END           'End of file reached.'
```

AW

Alternatives:

I

```
      20 STOP
      END
```

```
      20 END.
```

II Replace GO TO loop by a Do-loop.

```
Do      ! New statement
10  READ (1, *, END=20) SID, HOURS
    :
    :
    :
! GO TO 10
  ENDDO  ! New statement
```

```
Do  I = 1, 1000          ← loop counter
10  READ (1, *, END=20) SID, HOURS
    :
    :
    :
!   GO TO 10
    ENDDO
    :
    :
    :
```

! Safer than Do without counter — if only executes 1000 times, at most.
This assumes there are $\leq$ 1000 students in data file.

Older form:

```
Do  40  I = 1,1000
    :
40  CONTINUE    ! in place of ENDDO.
```

New form Do with label:

```
Loop_name :  Do I = 1,1000
    :
    :
ENDDO Loop_name
```

Named loops are easy to distinguish.
Useful with lots of Do loops.

(III) ↑ Part(iii) of question

We use a counter for each class of student.

```
INTEGER :: ifr = 0, iso = 0, iju=0, ise=0, iun=0
    :
```

New IF block:

```
IF (0<HOURS .AND. HOURS <= 30) THEN
    CLASS = 'Freshman'
    ifr = ifr + 1
ELSEIF (30<HOURS .AND. HOURS <= 60) THEN
    CLASS = 'Sophomore'
    iso = iso + 1
ELSEIF ...
    :
ENDIF
```

Add code to output results:

(IV) Replace the IF - block by a CASE - block.

Part (iv)

```
SELECT CASE (HOURS)

CASE (1:30)
    CLASS = 'Freshman'
CASE (31:60)
    CLASS = 'Sophomore'
CASE (61:90)
    CLASS = 'Junior'
CASE (91:)
    CLASS = 'Senior'
 CASE DEFAULT
    CLASS = 'Unknown'
END SELECT
```