Fortran Assignment.

1) Explain different data type in fortran.

Generally speaking, there are 5 data types in fortran

a) Integer
b) Real
c) Complex
d) logic
e) Character

Integer

eg: Integer a
a = 1

Real

eg: Real b
b = 1.2

Complex

eg: Complex c
c = 1,2.

logical

It contains two value of logical constant
-> TRUE.
-> FALSE

Character -> It contain the alphabets stored in ASCII value

Explain the Fortran structure

Generally, A structure means a template for a record. The name of a structure is defined & it can be used record statement.

Consider a example

```
STRUCTURE / PRODUCT /
    INTEGER ID
    CHARACTER NAME(100)
END STRUCTURE
```

In above example a structure named product is defined which consists of ID & Name.

Syntax.

```
structure [/structure Name/]
        Member of structure Declaration
END STRUCTURE
```

Logical If

1) Syntax [IF (condition statement)] 2) It is used to transfer the logical condition & transfer control depending upon the the control accordingly value of an expression.

3) eg:- IF (I.GT.2) SUM=SUM+1

Arithmetic if.

1) Syntax [IF (Expression) N_1, N_2, N_3] IF the condition is true N_3. where statemat is excited otherwise Expression = Valid arithmetic expression control & transfer gose to N_1, N_2, N_3 = statement number next statement.

2) It is used to check any given

3) Eg: IF (3*x+2) 10, 30, 25.

Characteristics of Fortran.

→ FORTRAN produces efficient codes
→ It is portable
→ It is very straight forward to learn & use.

Condition.d (Computed) Unconditional gotoes.

i) It is used to transfer control depending upon the ~~the control~~ condition

ii) It is used to transfer control unconditionally

a) Syntax
GOTO (N1, N2, ...), V.
where,
V = integer variable
N1, N2 are statement no.

eg: goto (111, 511, 200), Y.

(b) Syntax
GOTO N1.
where
N1 is the statement no. to which the control must be transfered

eg: goto 0.

Q. Structure of a fortran program
A FORTRAN program consists of a main program, possible followed by one or more program (sub-program).

The structure is.

Main Program.

Program statements
declarations
statements
end.

Sub program
subroutine or function
declaration
statements
end.

Do loop.

i) It is used whenever a particular job is to be repeated no. of time.

For example

```
integer i, N, Sum
sum = 0.
Do 10, i = 10, N
   Sum = sum + i
   write (*,*) 'i = ' i
   write (*,*) 'sum = ', sum.
10 continue.
```

Implied Do loop

i) It is used manually in Array & is used only for the input & output.

For example

```
integer A(50), N.
write (*,*) 'Enter total Number'
read (*,*) N
Write (*,*) 'Enter value
read (**) (A(i), i=1,N)
Do 100 I = 1, N-1
DO 200 J = I+1, N,
IF (A(I) .LT. A(J)) Then
K = A(J)
A(J) = A(I)
A(I) = A(K)
END IF
200 continue
100 continue
write (**) (A(I) K=1,N)
stop
```

IMPLIED DO LOOP

Implied Do loop provide a fast way of listing many item. These item depending upon the place where implied Do is used can be variables expression or even implied do loop.

Syntax.

(N1, N2 --- Nn, DO-var = initial, Final, step)

Example

in the following the Do variable i can here value -1, 0, 1 & 2. The item to be listed is i.

(i, i = -1, 2)

→ for i = -1
the item listed is -1

→ For i = 0
the item listed is 0

→ For i = 1
the item listed is 1

→ For i = 2
the item listed is 2.

Combining these case, the items listed by the given implied Do are

-1, 0, 1, 2.

Unformatted Input / Output statement

The user can be design the output & input formate according to the need but when an unformatted input/output statement are used, the compile itself, deal the formate.

When the data are to be inputted or result is to be shown. We fully mention the data type & size. Those specification is called format specification.

general form of format statement.

n format ($s_1, s_2, \ldots$)

where n is the statement number

$s_1, s_2, \ldots$ are format specification

Example

Say you have an integer variable & you want to print 4 character a real number you want to print is in fixed point notation with 3 decimal place.

write (*, 900) i, x

you format (I4, F5.3).

The format label 900 is taken arbitrarily, After the keyword format follow the format code The code I4 is integer for width. F5.3 is the number printed with width 5 & 3 decimal place.

Fortran Constants.

Constants are fixed values & are generally input value. so, A number or string of fortran character is called constant (fortran)

Type,

(a) Integer 1, 2, 3, 4

(b) Real 1.1, 0.5

(c) Complex (1,2) (a+ib) + (a,b)

(d) logical True / false

(e) character 'AB'

Fortran Variables

They are the name of memory location whose value changes according to the processing Type

(a) Integer
(b) Real
(c) Complex
(d) Logical
(e) Character

Library function

These are the built in library fuction which can be directly used. They are supplied automatically by the system & can be used in expression in any program unit

eg. SQRT (x) → $\sqrt{x}$.

Computed go goto

* Syntax

goto (N1, N2, --), V

where,

V = Integer variable

N1, N2 ... = statement Number

Switch case

* Syntax

switch (expression)
{
case 1 :
{
statement ;
Break ;
}

default:
    statement;
    break;
}

2) It is used to transfer the control when the condition is meet & depending upon the condition, it transfer the control.

8) Switch statement in C tests the value of a variable & componit with multiple case. Once the case match is found the block of statement associated with that particular case is executed.

Quadratic Equation

```
real a,b,c,d,R, Real, Img, R1, R2
write (*,*) Enter value for a,b,c'
Read (*,*) a,b,c.

d = (b**2) - (4*a*c)
if (d.eq.0) then
        write (*,*) The roots are real & equal
        R = -b/(2*a)
        write (*,*) :Root =', R.

else if (d.gt.0) then
        write (*,*) The root are real & unequal
        R1 = (-b+sqrt(d))/(2*a)
        R2 = (-b-sqrt(d)/(2*a)
        write (*,*) The roots are ', R1, R2
else.
```

```
write (*,*) 'The roots are imaginary'
    d = -d
    Real = -b
    Img = sqrt(d)

write (*,*) 'Root1 = ', Real ' + ', Img, 'i'
write (*,*) 'Root2 = ', Real ' - ', Img, 'i'
end if
 stop
 end
```

Leap Year Check

```
integer Y.
 write (*,*) 'Enter any year'
 Read (*,*) Y
 if ((mod(Y,4)).eq.0) then
      if ((mod(Y,100)).eq.0) then
           write (*,*) 'Entered Year is leap Year.
 else.
           write (*,*) 'Entered Year is not leap Year'
  endif
 else write (*,*) 'Entered Year is leap Year'.
   end if
 else.
      write (*,*) 'Entered Year is not leap year'
 endif

 Stop
 End.
```

Largest Number Among Entered Number

```
integer N,i,L,Num.
i=0
L=0
write (*,*) 'How many Numbers'
read (*,*) N
write (*,*) 'Entere the Numbers'

do while (i.ne.N)
        Read (*,*) Num
        if (Num.gt.L) then
        L=Num
        endif
        i= i+1
end do.

write (*,*) 'Largest Number =',L

stop
end
```

$\pi = 4\left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \ldots\right)$ upto 25 terms

```
real N, i, sum, Term, Fsum, x, a
N = 25
sum = 0
i = 1
a = 0

do while (i.le.25)
        x = (2*i)-1
        term = (1/x) * (-1) **(a)
        sum = sum + term
        i = i+1
        a = a+1
end do
Fsum = 4 * sum.
write (*,*) 'The value of pie is ' Fsum

stop
end
```

Binary To Decimal

```
integer Num, Bin, Dec, Power, Rem

Dec = 0
Power = 1

write (*,*) 'Enter a value in binary'
read (*,*) Bin

Num = Bin
do while (Num.gt.0)
        Rem = mod (Num,10)
        Dec = Dec + (Rem * Power)
        Power = Power *2
        Num = Num /10
end do

write (*,*) 'Equivalent Decimal value is', Dec.

stop
end.
```

Sorting of Numbers (10 Numbers)

```
int Num(100), i, j, k, N

N=10
write (*,*) 'Enter 10 Numbers'
read (*,*) (Num(i), i=1,N)
do 100 i=1, N-1
    do 200 i=i+1, N
        if (Num(i) .gt. Num(j)) then
            k = Num(j)
            Num(j) = Num(i)
            Num(i) = k
        end if
    200 continue
100 continue

write (*,*) 'After sorting'
write (*,*) (Num(k), k=1, N).

Stop
end
```

Sorting of Number in Ascending order of N element

```
integer Arr(1000) i,j,k

write (*,*) 'Enter the value for N'
read (*,*) N
write (*,*) 'Enter value'
read (*,*) (Arr(i), i = 1,N)

Do 100 i = 1, N-1
    do 200 j = i+1, N
    if (Arr(i).gtb Arr(j)) then
      k = Arr(j)
      Arr(j) = Arr(i)
       Arr(i) = k.
   end if
     200 continue.
100 continue.

write (*,*) 'Element in assending order'
write (*,*) (Num(k), k=1,N)

stop
end.
```

Displaying largest & smallest number from the list of 10 elements.

```
integer Num(100) i, j, k, N
N = 10
write (*,*) 'Enter 10 Numbers'
      (*,*) (Num(i), i = 1, N).

do 100 i = 1, N-1
      do 200 j = i+1, N
            if (Num(i).gt. Num(j)) then
                  K = Num(i)
                  Num(j) = Num(i)
                  Num(i) = k
            endif
      200 continue
100 continue

write (*,*) 'Largest Number is'
write (*,*) Num(N).

write (*,*) 'Smallest Number is'
write (*,*) Num(1).

stop
end.
```

Array Insertion

```
integer Num(1000) i, N, p, v, c, x

write (*,*) 'Enter the value of N'
read (*,*) N

write (*,*) 'Enter value'
read (*,*) (Num(i), i=1,N)
c=N

write (*,*) 'Before making changes'
read (*,*) (Num(i), i=1,N)

write (*,*) 'Enter the Number that you want to enter'
read (*,*) v

write (*,*) 'Enter the position that you want to
                    ente at'
read (*,*) p
x=p
do 100 i=p,v
        Num(c+1)=Num(c
        c=c-1

100 continue
```

Num (p) = b
write (*,*) 'After Making changes'
write (*,*) (Num (i), i = 1, N)

stop
end.

Transpose of Matrix

```
integer mat(100,100), Mat T(100,100) i, j, m,n
write (*,*) 'Enter the order of Matrix'
read (*,*) m,n

write (*,*) 'Enter the element for matrix'
read (*,*) ((Mat(i,j), j=1,n), i=1,m)

write (*,*) 'Before Transpo
   do 100 i=1,m   write (*,*) (Mat (i,j), i=1,n).
do 100 continue

write (*,*) 'After transpose'

do 200 i=1,m
    do 300 j=1,n
        Mat T(i,j) = Mat (j,i)
    300 continue
200 continue

    do 400 i=1,m
        write (*,*) (Mat T(i,j), j=1,n).
    100 continue

stop

end
```