```

*******************************************
./exercise1/logical_test.f90

PROGRAM logical_test

  IMPLICIT NONE

  LOGICAL :: tst1=.TRUE., tst2=.TRUE., tst3=.FALSE.
  LOGICAL :: ans1,ans2,ans3

  ans1=tst1 .OR. tst2 .AND. tst3
  ans2=(tst1 .OR. tst2) .AND. tst3
  ans3=tst1 .OR. (tst2 .AND. tst3)


  PRINT*, ans1,ans2,ans3

END PROGRAM logical_test


*******************************************


*******************************************
./exercise1/logical_test2.f90

PROGRAM logical_test2

  IMPLICIT NONE

  REAL     :: height=1.85, weight=95.0
  INTEGER  :: age=55
  LOGICAL  :: drinker=.TRUE., employed=.TRUE., smoker=.TRUE.
  LOGICAL  :: test1,test2,test3

  If (age>50) Print*,"Over fifty"
  test1 = employed .AND. (age<45)
  test2 = smoker .AND. drinker .AND. (height<2.00)
  test3 = (.NOT. smoker .OR. (age<=55) .AND. (weight>50)) .AND. (height<=1.84)


  PRINT*,test1,test2,test3

END PROGRAM logical_test2


*******************************************


*******************************************
./exercise6/array_example.f90

PROGRAM array_example
!** Program to declare a 4 by 4 matrix and then
!** initialise such that each element is the sum of its row
!** index and twice its column index

  IMPLICIT NONE

  REAL, DIMENSION(4,4) :: array !*** Create 2d array to represent a matrix
  INTEGER :: i,j !*** Define two loop variables

  DO i=1,4              !*** Loop over rows
    Do j=1,4            !*** Loop over columns
      array(i,j)=i+2*j  !*** Set matrix value
    END DO
```

```
    PRINT*,"Row ",i,"=",array(i,:) !*** Print each row to screen
  END DO

END PROGRAM array_example

*******************************************


*******************************************
./exercise3/power3_loop.f90

PROGRAM power3_loop
!*** calculate the 3rd power of a for 1 <= a <= 8

  IMPLICIT NONE

  INTEGER :: a, b

  DO a = 1, 8
    b = a*a*a
    PRINT*,a,"to the power of three = ",b
  END DO

END PROGRAM power3_loop

*******************************************


*******************************************
./exercise2/quadratic_real.f90

!****************************************************************************
!**
!**  PROGRAM:  Quad
!**
!**  PURPOSE:  Examine a quadratic equation
!**
!****************************************************************************

PROGRAM quad

!**
!** Program to investigate a quadratic equation (y=a*x*x+b*x+c)
!**

  IMPLICIT NONE      !** Force explicit declaration of all variables

  !** Declare required variables
  LOGICAL :: check
  REAL :: a=-3,b=6,c=1 !** Define the quadratic a*x*x+b*x+c
  REAL :: ans1,ans2   !** Declare two variables to hold the two roots
  REAL :: xturn       !** Declare variable to hold the x value of turning point
  REAL :: yturn       !** Declare variable to hold the y value of turning point
  REAL :: discrim     !** Variable to validate quadratic
  REAL :: realpart    !** Hold real part of complex number
  REAL :: imagpart    !** Hold imaginary part of complex number


  PRINT*,"Program to find the roots of a quadratic equation"
  PRINT*,"a=",a," : b=",b," : c=",c

  check= .NOT. a==0  !** Set check to .TRUE. if a valid quadratic

  !**
  !** Section to calculate root(s) of the quadratic
  !**

  IF (check) THEN

    discrim=b**2-4*a*c !** Calculate the discriminant of the quadratic.

    IF (discrim > 0) THEN !** If real solutions exist then enter construct

      PRINT*,"There are a two unique roots"
      ans1=(-b+sqrt(discrim))/(2*a) !** Calculate the first root
      ans2=(-b-sqrt(discrim))/(2*a) !** Calculate the second root

      PRINT*,"Root One =",ans1  ! ** Output the first root to the screen
      PRINT*,"Root Two =",ans2  ! ** Output the second root to the screen

    ELSEIF (discrim == 0) THEN
      PRINT*,"There is a single repeated root"
      ans1=-b/(2*a) ; ans2=ans1
      PRINT*,"Root =",ans1
    ELSE
      PRINT*,"No Real roots to this quadratic"
      PRINT*,"Complex roots are"
      realpart=-b/(2*a)
      imagpart=SQRT(-discrim)/(2*a)
      PRINT*,"Root One =",realpart,"+",imagpart,"i"
      PRINT*,"Root Two =",realpart,"-",imagpart,"i"
    ENDIF
  ELSE
    PRINT*,"This is not a valid quadratic"
  ENDIF

  !**
  !** Section to calculate the quadratic"s turning point and it"s nature
  !**

  IF (check) THEN
    !** Calculate the quadratics turning point
    xturn=-b/(2*a)
    yturn=-b*b/(4*a)+c

    PRINT*,"Turning Point = (x,y) = (",xturn,",",yturn,")"

    !** Calculate Max or Min point of
    IF (a .LT. 0)  THEN
      PRINT*,"Turning point is a maximum"
    ELSE
      PRINT*,"Turning point is a minimum"
    ENDIF
  ENDIF

END PROGRAM quad


*******************************************


*******************************************
./exercise5/nested_loop.f90

PROGRAM nested_loop
!**
!** Program to demonstrate nested loops
!**

  IMPLICIT NONE

  INTEGER :: loop1,loop2,ans=0

  DO loop1=10,19
```

```
    DO loop2=21,30
      ans=ans+loop1*loop2
    END DO
  END DO

  PRINT*,"Answer is",ans

  !** Ans = 36975

END PROGRAM nested_loop

*******************************************


*******************************************
./exercise4/factorial.f90

PROGRAM factorial
!*** calculate the factorials of the integers 1 -> 8

  IMPLICIT NONE

  INTEGER :: a, b=1

  DO a = 1, 8
    b = a*b
    PRINT*,"factorial ",a," = ",b
  END DO

END PROGRAM factorial

*******************************************

```