Quiz 1 (Close book, no cheat sheet, 30 minutes)
Your paper will not be graded unless you endorse the following statement:
I have neither given nor received inappropriate assistance on this quiz.

Part I Multiple Choice (40 points, 10 points each)
Clearly write the letter corresponding to the correct answer in the boxes on the right. You can have multiple answers.

1. Incorrect placement of Fortran statements may cause compiling errors. Which of the following arrangements would cause failure compiling?
   A. Comments, Declaration, Termination
   B. Declaration, Comments, Execution, Termination
   C. Comments, Execution, Declaration, Termination
   D. Declaration, Execution, Comments, Execution, Termination
   C

2. Select the valid declaration of a variable
   A. float :: aVariable
   B. int :: aVariable
   C. real :: aVariable
   D. character, parameter :: aString='aVariable'
   C

3. Which of the following logical expression generate a logical true
   A. '100' ==100
   B. .not. ( 'A' == 'B' )
   C. cos(0) == 1 .or. abs(-1) == 0
   D. exp( sin(0.0) ) >= 1.0
   B C D

4. Which of the following statements is NOT a valid Fortran 95/2003 variable name
   A. totalScore
   B. total Score
   C. total_score
   D. totalscore
   B
Part II Interpreting Fortran programs (30 points, 10 points each)

5. Read the following statements, and predict what will be printed to the monitor.

```fortran
integer::variable1 = 10
character(64):variable2 = ' == '
real::variable3 = 10
write (*,*) variable1, variable2, variable3, 
        " is true"
```

```
10 == 10.000000 is true
```

6. Read the following Fortran 95/2003 program clamp.f95. Please circle and correct the “buggy” statements. A “buggy” statement may cause compiling error, runtime error, or logical error.

```fortran
1 ! This program clamps a value into a
2 !   bounded range
3 program clamp
   ! A named constant needs to be assigned
4   real, parameter :: lowerThreshold = 0.0
5   real :: val = 0.0
6   write(*,*) "Please enter a number:"
7   read(*,*) val
8   if ( val < lowerThreshold ) then
9       val = lowerThreshold
10   end if
11   write (*,*) "The clamped value is: " val
12 end
```
7. Read the following pseudocode

```
get pFG value from keyboard
get pBG value from keyboard
get alpha value from keyboard

if alpha < 0. then
    alpha = 0.
else if alpha > 1. then
    alpha = 1.
end if

pBlend = pFG * alpha + pBG * (1-alpha)

if abs(pBlend) <= 1. then
    show the value of pBlend on the monitor
else
    show a failure message on the monitor
end if
```

A file named alphaBlend.f95 was written to translate the above pseudocode into a compiled Fortran 95/2003 program. Now you need to compile this program into an executable file and test it. Please predict the results that would be shown on the monitor given the following inputs from the keyboard.

<table>
<thead>
<tr>
<th>Inputs From Keyboard</th>
<th>Output on the Monitor</th>
</tr>
</thead>
<tbody>
<tr>
<td>alpha</td>
<td>pFG</td>
</tr>
<tr>
<td>-1.1</td>
<td>0.5</td>
</tr>
<tr>
<td>0.5</td>
<td>0.6</td>
</tr>
<tr>
<td>2.0</td>
<td>1.4</td>
</tr>
</tbody>
</table>

#1: alpha of -1.1 clamped to 0.0, then
  0.5*0.0+1.0*(1-0.0) = 1.0

#2: alpha of 0.5, no clamping, then
  0.6*0.5+0.2*(1-0.5) = 0.4

#3: alpha of 2.0, clamped to 1.0, then
  1.4*1.0-2.0*(1-1.0) = 1.4 > 1, 'show failure'
Part III. Writing Fortran 95/2003 programs (30 points, 10 points each) For all programs, you can ignore the heading comment block except a comment of the problem statement.

8. In many computer applications, high-cost computations such as exponential, logarithm, and trigonometric functions need to be replaced by the partial summations of their corresponding Taylor series to approximate the results with reduced computational cost. Please implement the following function using a second order Taylor expansion for fast approximation.

\[ y(x, u) = e^{-(x-u)^2} \]

You can use the following equation for 2\textsuperscript{nd} order Taylor expansion of the exponential function:

\[ e^x \approx 1 + x + \frac{x^2}{2} \]

Your program needs to get the user inputs for the values of \( x \) and \( u \), and output the result of \( y(x, u) \).

! A fast and low cost implementation of the function
! \( y(x, u) = \exp(- (x-u)^2) \)

program taylorApprox
  implicit none
  !declare input
  real:: x = 0., u = 0.
  !declare output
  real:: y = 0.
  !Get user input for x and u
  write(*,*) "Please enter the values of x and u:"
  read(*,*) x, u
  !Calculate the power
  x = -(x-u)**2
  !Apply the 2nd order Taylor expansion
  y = 1 + x + x**2/2.0
  !Output the result
  write(*,*) "The result is", y
end
9. In chess, the distance between squares on the chessboard for kings and queens is measured in Chebyshev distance, which can be calculated as follows:

\[ D_{\text{chebyshev}} = \max(|x_1 - x_2|, |y_1 - y_2|) \]

where \((x_1, y_1)\) is the chess board position of a king, and \((x_2, y_2)\) is the chess board position of a queen. Please write a Fortran program given the chessboard location of a king and a queen, output the Chebyshev distance between the two. Note: Please select the proper data type to hold the position of the chesses on the chess board. Please also clarify the coordinate of the chess board.

!Calculate the Chebyshev distance between a queen and a king
!on a chess board

```fortran
program chessDistance
    implicit none
    integer, parameter :: chessBoardSize = 8
    integer :: x1 = 0, y1 = 0
    integer :: x2 = 0, y2 = 0
    integer :: dist = 0

    write(*,*) "The origin of the chess board is at & the lower left corner."
    write(*,*) "Enter the position of the king:"  read(*,*) x1, y1
    write(*,*) "Enter the position of the queen:"  read(*,*) x2, y2

    !check if they are still inside the board
    if ( (x1 < 0 .or. x1 >= chessBoardSize ) .or. &
         (y1 < 0 .or. y1 >= chessBoardSize ) .or. &
         (x2 < 0 .or. x2 >= chessBoardSize ) .or. &
         (y2 < 0 .or. y2 >= chessBoardSize ) ) then
        write(*,*) "At least one of the rooks is outside the board"
        write(*,*) "Both coordinates need to be in the range & from 0 to ", chessBoardSize
        dist = -1
    else
        dist = max( abs(x1-x2), abs(y1-y2) )
    end if

    write(*,*) "The Chebyshev distance between & the king and the queen is ", dist
end
```
10. Running multiple commands to test different input configurations can be tedious. One method to make the program more intelligible is by adding user controls to the input configurations of the program. Please write a program to calculate the equivalent resistance value given 3 input resistances from user input, the user can control how these resistors are connected (in series, or in parallel), your program will display the corresponding display given the user input of their connection method. You don't need to simulate resistance noise. All resistance values are in the unit of Ohm. For components with resistances $R_1$, $R_2$, and $R_3$, the equivalent resistance in series can be calculated as follows:

$$R_{\text{series}} = R_1 + R_2 + R_3,$$

and the equivalent resistance in parallel can be calculated as follows:

$$R_{\text{parallel}} = \frac{1}{\frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3}}.$$

! Given three resistances, calculate the equivalent resistance according the user input of how they are connected (in series or in parallel)

```fortran
program threeResistors
  implicit none
  !declare input
  real :: r1=0., r2=0., r3=0.
  character :: connection = 'S'
  !declare output
  real :: rTotal

  !get input data
  write(*,*) "Please enter the three resistances:"
  read(*,*) r1, r2, r3
  write(*,*) "How do you want to connect them?"
  write(*,*) "Enter 'S' for connecting in series"
  write(*,*) "Enter 'P' for connecting in parallel"
  read(*,*) connection

  !Branch the equivalent resistance calculation according to the connection value
  if ( connection == 'S' ) then
    rTotal = r1 + r2 + r3
  else
    rTotal = 1./( 1./r1 + 1./r2 + 1./r3 )
  end if

  write(*,*) "The total resistance is ", rTotal
end
```