Homework Assignment 3

Section___________________

Student ID_________________________Name (Print)__________________________

Fundamentals of loop construct

Chapman textbook problems 4-25, 4-26, and 4-28
Due 10/13/08 11pm for section 1 and 2, due 10/14/08 11pm for section 3.
Please submit your source code using blackboard --> Assignments -->
View/Complete Assignment: Assig4_Fundamentals of loop --> Browse --> Save --> Submit

Late submissions will be counted off from your score by 1 point per hour.
Solution

4-25) **Decibels**: Engineers often measure the ratio of two power measurements in *decibels*, or dB. The equation for the ratio of two power measurements in decibels is

\[ dB = \log_{10}\left(\frac{P_2}{P_1}\right) \]

where \( P_2 \) is the power level being measured, and \( P_1 \) is reference power level (1 milliwatt). Write a program that calculates the decibel level corresponding to power levels between 1 and 20 watts, in 0.5-W steps.

**Source code: 4-25.f95**

```fortran
!Chapman 4-25 insert do-while loop based on 2-22.f95

program decibels
    implicit none
    real, parameter::p1=1
    real::p2=0, p2dB=0.
    p2=1.0
    do while ( p2 <= 20.0 )
        p2dB = 10*log10(p2/p1)
        write(*,*) p2, p2dB
        p2 = p2+.5
    end do
end program
```
4-26) **Infinite Series** Trigonometric functions are usually calculated on computers by using a truncated infinite series. An infinite series is an infinite set of terms that together add up to the value of a particular function or expression. For example, one infinite series used to evaluate the sine of a number is
\[
\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \ldots
\]
or
\[
\sin x = \sum_{n=1}^{N} (-1)^{(n-1)} \frac{x^{2n-1}}{(2n-1)!}
\]
where \(x\) is in units of radians.

Since a computer does not have enough time to add an infinite number of terms for every sine that is calculated, the infinite series is truncated after a finite number of terms. The number of terms that should be kept in the series is just enough to calculate the function to the precision of the floating point numbers on the computer on which the function is being evaluated. The truncated infinite series for \(\sin x\) is

where \(N\) is the number of terms to retain in the series.

Write a Fortran program that reads in a value for \(x\) in degrees, and then calculates the sine of \(x\) using the sine intrinsic function. Next, calculate the sine of \(x\) using Equation 4-13, with \(N = 1, 2, 3, \ldots, 10\). Compare the true value of \(\sin x\) with the values calculated by using the truncated infinite series. How many terms are required to calculate \(\sin x\) to the full accuracy of your computer?
Planning the implementation:
Top-down design strategy to solve this problem

Flowchart of the main program

```
start
Get user input for x in degree
Convert x from degree to radius
SinIntrinsic = sin(x)
N=1, N<=10, N=N+1
mySin = sinApprox(x, N)
Write(*,*) N, mySin
end
```

Flowchart of the function `sinApprox`

```
Start argument x, N
sinApprox = 0.
i=1, i<=N, i=i+1
sinApprox = sinApprox + (-1)**(i-1) * x**(2*i-1) / fact(2*i-1)
end
```

Flowchart of the function `fact`

```
Start argument x
fact = 1.
i=1, i<=x, i=i+1
fact = fact * i
end
```
program testSinApprox
    implicit none
    real, parameter::PI = 3.1415926
    integer::totalIter=0
    real::x=0.
    real::sinApprox

    write(*,*) "Enter an angle in degree:"
    read(*,*) x
    x = x * PI / 180.

    do totalIter = 1, 10
        write(*,*) "IntrinsicSin=", sin(x), &
            "MyApproximation=", sinApprox(x, totalIter), &
            "N=", totalIter
    end do
end program

real function sinApprox(x, totalIter)
    implicit none
    real, intent(in)::x
    integer, intent(in)::totalIter
    integer::factorial

    integer::n=0,m=0
    sinApprox = 0

    do n = 1, totalIter
        m = 2*n-1
        sinApprox = sinApprox + (-1)**(n-1) * x**m / factorial(m)
    end do
end function

integer function factorial( val )
    implicit none
    integer,intent(in)::val

    integer:: i
    factorial = 1
    do i = 1, val
        factorial = factorial * i
    end do
end function
4-26) **RMS Average** The *root-mean-square (rms)* average is another way of calculating a mean for a set of numbers. The rms average of a series of numbers is the square root of the arithmetic mean of the squares of the numbers

\[
rmsAverage = \sqrt{\left( \frac{1}{N} \sum_{i=1}^{N} x_i^2 \right)}
\]

Write a Fortran program that will accept an arbitrary number of positive input values and calculate the rms average of the numbers. Prompt the user for the number of values to be entered, and use a DO loop to read in the numbers. Test your program by calculating the rms average of the four numbers 10, 5, 4, and 5.

Solution source code 4-26.f95

```fortran
!Chapman 4-28 rms average
program rmsAverage
    implicit none
    real::x=0, rms=0
    integer::N=4,i=0

    rms=0
    i=1
    do
        if ( i > N ) then
            exit
        end if
        write(*,*)"Enter number ", i, ":"
        read(*,*) x
        rms = rms + x*x
        i = i+1
    end do
    rms = sqrt( rms / N )
    write(*,*) "The rms average of your input numbers is: ", rms
end program
```

```
./a.out
Enter number 1 : 10
Enter number 2 : 5
Enter number 3 : 4
Enter number 4 : 5
The rms average of your input numbers is: 6.442050
```