CSE 121 -- Introduction to "C" ***(Worth 15 pts)***

Extra Credit #3 (extra3.c) Due: last day of class at 11:59PM

Name your program: extra3.c

In this extra credit assignment, you'll write your own square-root function that corresponds to the following prototype:

double mySqrt (double x); (for $x \geq 0$)

Now, computers can only perform: +, -, *, /, and do some comparisons. So for any mathematical function (e.g. tan, sin, square-root, log,.....) to be programable on a computer, you need to find a way to calculate the answer that only uses: +, -, *, /, and some comparisons. For a square-root function you can use the *Newton-Raphson* method. The Newton-Raphson method is a general method that will find the value of x that makes a function f(x) = 0. (under the constraint that the function f(x) is differentiable).

In our case, we are trying to determine x for the equation: $x = \sqrt{a}$

Which can be rearranged to: $x^2 = a$

and finally in standard polynomial form: $x^2 - a = 0$

So our function f(x) = 0, can be written as: $f(x) = x^2 - a = 0$

The Newton-Raphson equation for iteratively solving an equation f(x) = 0 is:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

Where for our case:

$f(x_k) = x_k^2 - a$

$f'(x_k) = 2 \cdot x_k$ (i.e. the derivative of f(x))

After substituting f(x) and f'(x) into the Newton-Raphson equation and simplifying, we have:

$$x_{k+1} = 0.5 \cdot \left(x_k + \frac{a}{x_k} \right)$$

$$x_{k+1} = 0.5 \cdot \left(x_k + \frac{a}{x_k} \right)$$

So to use the above equation to find the $\sqrt{a}$ you must pick an initial value for x_k at k = 0 (i.e. the beginning of the first iteration)

A reasonable initial guess for the square-root of "a" is always: $\frac{a}{2}$

So let $x_0 = \frac{a}{2}$

So plug the initial value x_0 into the equation above and find x_1 , then plug x_1 into the equation and find x_2 . Continue to iterate the equation until x_k and x_{k+1} are nearly the same (i.e. the iteration has converged to an answer). So what does x_k and x_{k+1} are nearly the same mean ? It means you must iterate the equation until the difference between them is less than some relative error. So let the relative error be:

$$\left| \frac{x_{k+1} - x_k}{x_{k+1}} \right| < 0.001$$ (i.e. error is less than 1/10 %)

USE-CASE DESCRIPTION
1. ***User*** --- types ./extra3 2. ***Application*** --- displays on the screen, "Find the square root of:" 3. ***User*** ---- enters a number and hits return. 4. ***Application*** --- Prints the square root of the number entered, or an error message if the user enters a negative number.
NOTES AND TBDs
- Your function should check to make sure the number entered by the user is not negative. If the user enters a negative number and that negative number is given to the function mySqrt(x), mySqrt(x) should return -1 as an error code. Your main program can check for the error code and decide what message to print to the user if the error code is returned. - To calculate the relative error, you will need an absolute value function. You can write your own, or use the fabs( ) function from the #include <math.h> library, but if you use it, you must include the -lm option when compiling: gcc extra3.c **-lm** -o extra3

Use the following command to submit your extra3.c code

cp extra3.c /home/faculty/skoss/cse121/your_UID