Assignment 04

Due: Monday, Feb. 20, 2017 at noon 100 pts

For this assignment, you will submit a single C++ compilable file containing a program written in C++. Remember, to submit a file for this course electronically, from the directory in which the file resides type in at the UNIX prompt the command: cssubmit 1570 section_letter assignment_number. Be sure that only the file you want to submit is in that directory - make a directory for every assignment! The submit system will deliver every .cpp file in the current directory to me. Name your file a meaningful name and give it a .cpp extension since you will be compiling it. Also, make sure that you compile and run your program using the GNU (g++) compiler before submitting to make sure that it will work for the submit script.

Background: Your last assignment was to write a program to help out our hero, Milhouse, with his math skills....well, help him to *ignore* his lack of math skills. Now Milhouse wants to improve those skills, so you are going to write a program that will help him check his work. He will be manually computing values for exponentials, trig functions, and roots of numbers. So he needs a "calculator" of sorts so he can check his answers.

Specifications: Your program will begin by presenting a menu with these choices:

OPTIONS
---

1. Exponential of x
2. Sine of x
3. Roots of x
4. Quit (and run away)

In fact, your program should present this menu until the user chooses the quit option.

When option #1 is chosen, the user is to be prompted for the value of x that the user wishes to exponentiate. Accept only values (real numbers) between and including 0 and 7. You are to use this formula for the computation:

$$e^x = 1 + x + x^2/2 + x^3/6 + x^4/24 + x^5/120 + x^6/720.$$

That's an approximation that should be pretty good. Output the computed value.

When option #2 is chosen, the user is to be prompted for the value of x (any real value) for which to compute the sin(x). But then, they are to next be prompted for a positive integer between and including 1 and 5 representing the accuracy of the computation. This value will determine the number of terms to be included in the truncated Taylor series which calculates an approximation to the sin(x). This is that formula:

$$\sin(x) = x^1/1! - x^3/3! + x^5/5! - ... + x^{2k-1}/(2k-1)! \qquad k = 1, 2, 3, ...$$

Thus, if one wants 2 terms, that corresponds to k = 2 in this formula and $\sin(x) = x^1/1! - x^3/3!$. Of course, it's not really equal (=), but we'll pretend. By the way, you will learn in calc II that the larger k is, the better the approximation is. And if you add on terms forever, the equality is indeed guaranteed. Try it! It's definitely worth the time. Note: k! is "k factorial" is k * (k-1) * (k-2) * ... * 3 * 2 * 1 (e.g. 4! = 4*3*2 *1 = 24). And, of course, output the value.

When option #3 is chosen, you will present the user with another menu that looks like this:

Roots
---
1. enter x
2. square root
3. cube root

Here, options 2 and 3 should be denied until option 1 is first chosen. Once option 1 has been chosen, options 2 and 3 will compute and output the square root of x and the cube root of x, respectively. After the value is output, return to the first menu. Now, how are you to compute the roots? You will code an iterative method (Newton's method) to compute the roots of x. What that means is that you compute a new x value based on the old x value. Here's the formula for the square root:

$$x_{n+1} = (x_n + A/x_n) / 2 \qquad \text{for } n = 0, 1, 2, ...$$

So, A is the number you are trying to find the square root of (e.g. square root of 16 is 4 since 4^2 is 16), x_0 is an initial guess, and

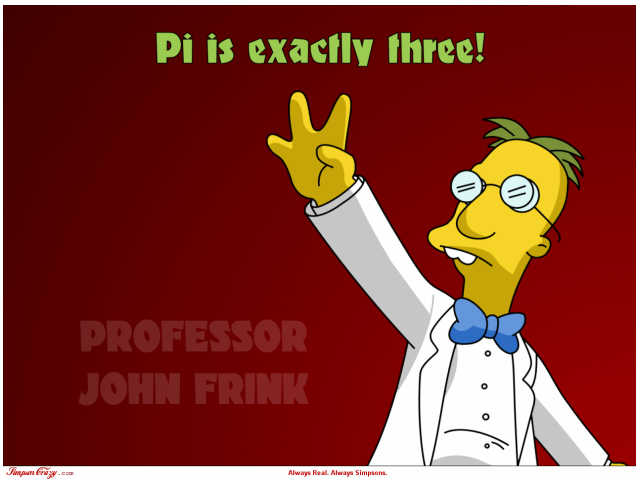
$x_1 = (x_0 + A/x_0) / 2,$
$x_2 = (x_1 + A/x_1) / 2,$
$x_3 = (x_2 + A/x_2) / 2,$
etc.

If you keep doing this forever, the last x value you compute is the square root of A. Cooool? Well, yeeaaahhhh, sort of. You don't have infinite time, hence you cannot iterate forever. But, Newton's method "converges" to a value very close to the answer really quickly no matter what your first guess is. So, you will code it to repeat this computation just 6 times and use an initial guess (that's x_0) of A, that's just the value you are trying to find the square root of. (And to be sure you understand, A is the value you prompt for in option 1 of option 3.) Try this with your calculator: in the formula above, let A be 16 and start with 16 as an initial guess at the square root. You'll see that the x_n's get really close to 4 pretty fast. And for computing the cube root of A, use

$$x_{n+1} = (2x_n + A/x_n^2) / 3 \qquad \text{for } n = 0, 1, 2, ...$$

Details: For this assignment, you are to use the switch-case statement to branch to the various choices for the first menu. You are forbidden to use <cmath> (which is the same as <math.h>) for this assignment! If you want to confer with Professor Frink, that's ok. Your code had better make sure that allowable values are entered upon prompts. And as far as we are concerned, you can only find the roots of positive numbers. Also, think about what kind of precision you might want in your computations for the formulas above.

When you submit: the submit script will (attempt to) compile and execute your program in the process. This means that you will be the "user" of the program for that few minutes. Now, in order that the grader isn't driven crazy by a multitude of inputs/outputs, you will ALL input the same values so that he has uniform output to grade. They are:



- choose a nonexistent menu option such as 5 (which should generate a "invalid input error message")
- choose option #1
- enter x = -3
- enter x = 3
- choose option #2
- enter x = 0.5
- enter number of terms to be 3
- choose option #2
- enter x = 0.5
- enter num terms to be 5 (marvel at the accuracy!)
- choose option #2
- enter x = 0.5
- enter num terms to be 9
- enter num terms to be 1
- choose option #3
- choose option for square root (which should generate an error)
- choose option 1 and enter x = 27
- choose option for square root
- choose option #3 again and enter 27 for x
- choose option for cube root
- quit

As usual, if you have issues with this assignment, don't hesitate to ask your cs 1570 instructor.

E-mail: price@mst.edu | Phone: 573-341-4491 | Fax: 573-341-4501 | Address: 325G Computer Science Building, Rolla, MO 65409-0350