**ENGR 102**

**Lab 6B [100 Points]**

**Program #1: Taking limits to compute derivatives [50 Points]**

In an earlier lab, we observed how we could have a function that is undefined at some value (such as (sin x)/x at the point x=0), but could come arbitrarily close to it by successively evaluating smaller and smaller numbers (i.e. taking a limit). For example, we might evaluate at x=0.1, x=0.01, x=0.001, etc. until we have come very close to the value. Taking limits like this, numerically, is commonly done when functions are too complicated to evaluate analytically. You will write a program to compute a derivative as a numerical limit. This activity has a few parts:

You may reuse code from earlier activity if it is helpful.

1. Evaluating a polynomial limit analytically **[10 Points]**

You should have learned by now the process for finding the derivative of a polynomial (as another polynomial). Write a program that will read in from the user a cubic polynomial *f(x)* (as a set of 4 coefficients), and use this to compute the derivative polynomial (i.e. compute the three coefficients of the derivative *f’(x)*). Then, read in a value for *x* from a user, and evaluate the derivative polynomial at that x. Print out that value.

1. Evaluating a polynomial derivative numerically **[20 Points]**

For a function *f(x)*, the derivative of the function at a value *x* can be found by evaluating and finding the limit as *a* gets closer and closer to 0. Using the same polynomial as the user entered in part (a), and for the same value of *x* as entered in part (a), compute the limit numerically. That is, start with an estimate by evaluating using a value for *a* such as 0.1. Then, repeatedly halve the value of a until the difference between successive evaluations of is less than some small value, such as 10-6. Print the result, along with the number of evaluations it took. Calculate how close that result is to the actual answer, computed in part (a).

**Challenge:** Derivatives can also be estimated by computing the limit or . Try computing each of those, and calculate how many iterations you need to converge to the limit. Do you get different results with any of them, or does any of them take fewer steps to get an answer?

1. Evaluating a more complex function. **[20 Points]**

In your own code, come up with four more complex functions (not a polynomial – e.g. use sin/cos/tan/exp/log/powers/etc.), that you do not know how to compute the derivative for analytically, but that you can evaluate. For each function, using the same process as in part (b), calculate the derivative of that function at some value. Write a line of output describing each function, and stating what the computed derivative for it is, along with the number of steps needed to compute the derivative.

Be sure to include appropriate comments in your code, and to use descriptive input and output statements.

**Activity #2: Write programs to do each of the following. The programs should be labeled 2a through 2c: [50 Points]**

For all programs, you should Include comments in your code that describe the purpose of individual blocks.

1. The Collatz conjecture: **[25 Points]**

The Collatz conjecture, also known as the 3n+1 conjecture (and other names), deals with the following operation to produce a sequence of numbers. Given a number, n, if n is even then the next number is n divided by 2. If n is odd, then the next number is 3n+1. The Collatz conjecture is that this sequence of numbers always eventually reaches 1. As simple as this seems, it is unproven (and considered extremely hard to prove) by mathematicians.

As an example of a sequence, if you start with the number 6, then the terms of the sequence will be: 6, 3, 10, 5, 16, 8, 4, 2, 1.

Write a program that takes in a positive integer from a user, and computes the Collatz sequence, printing out all the numbers in the sequence, followed by a line stating how many iterations it took to reach the value 1.

1. Averaging measurements: **[10 Points]**

Assume that someone has collected a set of measurements and wants some statistical data about them. Write a program that asks a user for measurements and prints the average, the maximum, and the minimum measurement. Users should be allowed to enter as many measurements as they want, until entering a negative measurement. The negative measurement should not be processed, but is just used to indicate that the user has finished entering measurements. [*Note: do not use a list to store the measurements*.]

1. Divisors: **[15 Points]**  
   For numbers from 2 to 100, print a series of lines indicating which numbers are divisors of other numbers. For each, print out “X divides Y”, where X <= Y, and both X and Y are between 2 and 100. The first few lines will be:

2 divides 2

3 divides 3

2 divides 4

4 divides 4

5 divides 5

etc.