Caterin Duarte Reynosa
cduarter@ucsc.edu

CSE13s Fall 2020
Assignment 2: A Small Numerical Library
Design Document

Goal

The goal for this assignment is to create 5 different math 'functions' without using c's math library. This will be done by using Taylor Series and Newton's method. We will also be creating test harnesses in order to test how accurate our implementations are. I will be showing my accurateness and how my own functions compare to the math libraries functions. In order to test sin and cos, both functions will be tested in the range −2π to 2π. Tan will be tested in the range of −π/3 to π/3. e **x and log will be tested in the range from ranges 1 to10.

Things I have to do:

- Create a Makefile
  - Must support make, make all, build mathlib-test programs, make clean must remove binaries
- Create a README.md
- Create a WRITEUP.pdf
  - What I learned, what my code does, and graphs comparing my code to the real functions
- Use mathlib.h
- Create mathlib-test.c
  - accept : -a: to run all tests, -s: to run sin tests, -c: to run cos tests, -t: to run tan tests, -e: to run e**x tests, -l: to run log tests.
- Create mathlib.c
  - Sin, Cos, Tan, E**x, Log

SINE

Taylor Series for Sin: $x - \frac{x-x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!}...$

Pseudocode:

- Create a counter to track numerator, have it start at 1
- Create a counter to track denominator, have it start at 1
- Have a counter that will return the actual sum, have it start at numerator/denominator
- Have a counter that tracks the 'factorial'
- Have a counter that tracks the term
- A loop that stops when the term is greater than 1e-14
- Increment the counter that tracks the factorial by 3
- Have my numerator counter be set to itself multiplied by the input
- Have my denominator counter be set to itself multiplied by the factorial -1, then the factorial counter again
- Have my term counter be set to numerator/denominator
- Add the term to the sum

- Return sum

COSINE

Taylor Series for Cos: $1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!}...$

Pseudocode:

- Create a counter to track numerator, have it start at 1
- Create a counter to track denominator, have it start at 1
- Have a counter that will return the actual sum, have it start at numerator/denominator
- Have a counter that tracks the 'factorial'
- Have a counter that tracks the term
- A loop that stops when the term is greater than 1e-14
- Increment the counter that tracks the factorial by 2
- Have my numerator counter be set to itself multiplied by the input
- Have my denominator counter be set to 1 mines itself multiplied by the -factorial
- Have my term counter be set to numerator/denominator
- Add the term to the sum
- Return sum

TAN

Return sin / cos

EXP

- Have the factorial counter be set to 0
- Have the denominator counter be set to 1
- Have the numerator counter be set to 1
- Have the term counter be set to numerator / denominator
- Have the sum counter be set to the term
- Have a loop that stops when term > epsilon
- Have the term be equal to the input divided by the factorial counter
- Have the term then be multiplied by the old term
- Increment the sum counter by by term
- Return the sum

Log

- Have a counter be set to 1
- Have a counter be set to y times itself
- Have a loop that stops when (input - second counter) > Epsilon
- Have first counter be set to itself plus ((input - second counter) / second counter))
- Have second counter be multiplied by itself again
- Return first counter
-