Assignment 2- A Small Numerical Library

This program implements functions that have been written for sin, cos, tan, e^x, and log and creates a small numerical library with these functions. Then, employing a test-harness, specific functions will be called to output with command-line options.

Basic Organization:

- In mathlib-test.c
    - defined command-line functions in a string where:
        - -a: all functions
        - -s: sin function
        - -c: cos function
        - -t: tan function
        - -e: e^x function
        - -l: log function

    - contains main():
        - calls math functions from mathlib.c
        - normalizes input values to functions if necessary
        - uses return values from mathlib.c to print output when specific getopt() options are called

          * allows any combination of options to be called— but there is no repeat outputs of tests

- In mathlib.c
    - functions for basic operations written i.e. factorial, power, etc.
    - functions for sin, cos, and exp. written
        - sin & cos: Taylor series centered @ 0
        - e^x: uses recursion
    - function for log uses Newton-Raphson's(iteration)
    - function for tan uses written sin and cos functions
    - epsilon (10^{-14}) halts computation

Taylor Series >> Simplified for code:

- **sin(x): (x^1/1!) - (x^3/3!) + (x^5/5!) - (x^7/7!)**

From one term(k) to the next(k+1):

Numerator = initial (x) multiplied by -x^2(negative to switch signs)

Denominator = initial (1) multiplied by (k+1)(k+2)

Term = Numerator/Denominator
Sum = initial sum (set to first term = x); but after is Sum = Sum + Term

- **cos(x): 1 - (x^2/2!) + (x^4/4!) - (x^6/6!) + (x^8/8!)**

From one term(k) to next(k+1):
Numerator (initial = 1), Denominator (initial = 1), Term, Sum = same as for sin
Only change is initial Sum = 1

- **tan(x)**:

No calculations required; just return sin/cos

- **exp(x): 1 + x + (x^2/2!) + (x^3/3!) + (x^4/4!) + (x^5/5!)**

From one term(k) to next(k+1):
Numerator = initial (1) multiplied by x (no negative because no sign switch)
Denominator = initial (1) multiplied (k+1)
Term = Numerator/Denominator
Sum = initial (1); but after is Sum = Sum + Term

- **log(x):**

From one term(k) to next(k+1):
Numerator = (x - exp(k)/exp(k)) to previous term; initial (1)
Denominator = exp(k); initial (1)
Term = Numerator/Denominator
Sum = initial(1); after is Sum + Term

Pseudocode:

1) mathlib.c:

```
function (sin)-
    set sum and num = x
    den = 1
    term = num/den
    for k = 1,
        if abs(term) > 10e-14 -> increment k by 2
            num = num * -x^2
            den = den * (k+1)(k+2)
            reset term to num/den
            sum = sum + term
        return value sum
function (cos)-
        set sum, num, den = 1
        term = num/den
        for k = 1
                if abs(term) > 10e-14 -> increment k by 2
                        num = num * -x^2
                        den = den * (k+1)(k+2)
                        reset term to num/den
                        sum = sum + term
        return value sum
function (tan)-
        return sin/cos
function (exp)- **did not work as intended— referred back to Prof.'s code**
        set num, den, sum = 1
        set term = num/den
        for k = 1
                if abs(term) > 10e-14 -> increment k by 1
                        num = num * x
                        den = den * (k+1)
                        reset term to num/den
                        sum = sum + term
        return value sum;
function (log)- **did not work as intended— referred back to Prof.'s code**
        set sum, num, den = 1;
        set term = num/den;
        while abs(exp(k) - x) > 10e-14
                num = (x - exp(k))/exp(k)
                den = exp(k)
                term = num/den
                sum = sum + term
                k = sum
```

2) mathlib-test.c:

```
for all variables that call function to run
        set boolean value to false
        for getopt options = 'asctel'
                if flag = a -> set bool = true
                if flag = s -> set bool = true
```

```
I       if flag = c -> set bool = true
        if flag = t -> set bool = true
        if flag = e -> set bool = true
        if flag = l -> set bool = true
        set default = error message
if all = true:
        execute all math functions
if sin = true:
        print header formatted with spacing and x values, own_sin function values,
library_sin values, and diff (own-library)
        while within period [-2pi, 2pi]
                execute sin
                if > 2pi or <-2pi
                        normalize to range
if cos = true:
        print header formatted with spacing and x values, own_cos function values,
library_cos values, and diff (own-library)
        while within period [-2pi, 2pi]
                execute cos
                if > 2pi or <-2pi
                        normalize to range
if tan = true:
        print header formatted with spacing and x values, own_tan function values,
library_tan values, and diff (own-library)
        while within period [-pi/3, pi/3]
                execute tan
if exp = true:
        print header formatted with spacing and x values, own_exp function values,
library_exp values, and diff (own-library)
        while within period [1, 10)
                execute exp
if log = true:
        print header formatted with spacing and x values, own_log function values,
library_log values, and diff (own-library)
        while within period [1, 10)
                execute log
```