asgn2: DESIGN.pdf - A Little Slice of π

Student name: *Serjo Barron*

Course: *CSE13S* – Professor: *Prof. Long*
Due date: *10/7/21*

1 Program

This program simulates a small number of mathematical functions ($e^x and \sqrt{x}$) included in the <math.h> library namely, and then uses those functions to compute the fundamental constants e and π. π specifically is calculated using 4 different formulas, each inspired by the methods of Euler, Baily-Borwein-Plouffe, Viete, and Madhava. The 6 implemented functions will then be compiled all into a single test harness, which will be where you can interact to compare the results of the mimicked <math.h> library, and the actual functions that are provided inside the <math.h> library. The test harness will be able to show the difference between the values computed, and the amount of iterated terms each function needed (for the mimicked <math.h> library).

2 Required Files

1. e.c
    - This file includes the implementation of the e() and e_terms() functions.
2. newton.c
    - This file includes the implementation of the sqrt_newton() and sqrt_newton_iters() functions.
3. euler.c
    - This file includes the implementation of the pi_euler() and pi_euler_terms() functions.
4. bbp.c
    - This file includes the implementation of the pi_bbp() and pi_bbp_terms() functions.
5. viete.c
    - This file includes the implementation of the pi_viete() and pi_viete_factors() functions.
6. madhava.c

- This file includes the implementation of the pi_madhava() and pi_madhava_terms() functions.

7. mathlib-test.c
    - This file will be the main testing harness for the above functions (e.c, newton.c, euler.c, bbp.c, viete.c, and madhava.c).
8. mathlib.h
    - This header file will be included in all ".c" files and allows for proper linking, and usage of the functions inside the test harness.
9. Makefile
    - A file that will link all functions to the test harness (mathlib-test.c) and compile them into an executable.
10. README.md
    - A text file in markdown syntax that will provide the instructions on how to run the program, and provide a brief program description.
11. DESIGN.pdf
    - The design process of the program (the current document you are reading).
12. WRITEUP.pdf
    - An analysis comparing the mimicked <math.h> library (which you are making), and the actual <math.h> library. This document will discuss and explain any discrepancies that are found in comparing the values of the two methods of calculating values (included with corresponding graphs).

3 Psuedocode, Structure, and corresponding Formulas for ".c" files

e.c : Euler's Number

$$\frac{x^k}{k!} = \frac{x^k - 1}{(k-1)!} \times \frac{x}{k}$$

```
define e function:
    for (k = 0, current_term > epsilon, increment k by 1):
        increment count by 1
        if on the first iteration (k is 0):
            total += previous_term (previous_term's initial value is 1)
        else:
            x_over_k (x/k) = 1 / k
            current_term = previous_term × x_over_k
            total += current_term
            previous_term = current_term

    return total (total is the computed value e)

define e terms function:
    return count
```

newton.c : Newton Raphson Method

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

```
(the pseudocode below was made using the python example of newton's sqrt method
provided by Prof. Long inside of the corresponding asgn 2 specifications/document)

define sqrt newton function: (parameters: takes a value x to be square rooted)
    count = 0 (used to reset the count to 0 for every sqrt computed)
    while the absolute value of (upper - lower) > epsilon:
        increment count by 1
        lower = upper
        upper = 0.5 × (lower + (x / lower))
    return upper

define sqrt terms function:
    return count
```

euler.c : Euler's Solution

$$p(n) = \sqrt{6 \times \sum_{k=1}^{n} \frac{1}{k^2}}$$

```
1  (this function makes use of the sqrt function provided in newton.c, which will be
2  referenced as sqrt() in this pseudocode)
3
4  define pi euler function:
5      for (k = 1, current_term > epsilon, increment k by 1)
6          increment count by 1
7          exponent = k × k
8          current_term = 1 / exponent
9          total += current_term
10     total = 6 × total
11     result = sqrt(total)
12     return result
13
14 define pi euler terms function:
15     return count
```

bbp.c : Bailey-Borwein-Plouffe's Formula

$$p(n) = \sum_{k=0}^{n} 16^{-k} \times \frac{(k(120k+151)+47)}{k(k(k(512k+1024)+712)+194)+15}$$

```
1  define pi bbp function:
2      for (k = 0, current_term > epsilon, increment k by 1) (initial value of current_term is 1)
3          increment count by 1
4          fraction1 = 1 / exponent (initial value of exponent is 1)
5
6          top = (k × ((120 × k) + 151) + 47)
7          bottom = (k × (k × (k × ((512 × k) + 1024) + 712) + 194) + 15)
8          fraction2 = top / bottom
9
10         current_term = fraction1 × fraction2
11         total += current_term
12         exponent ×= 16
13     return total
14
15 define pi bbp terms function:
16     return count
```

viete.c : Viète's Formula

$$\frac{2}{\pi} = \frac{\sqrt{2}}{2} \times \frac{\sqrt{2+\sqrt{2}}}{2} \times \frac{\sqrt{2+\sqrt{2+\sqrt{2}}}}{2} \times ...$$

```
1  (this function makes use of the sqrt function provided in newton.c, which will be
2  referenced as sqrt() in this pseudocode)
3
4  define pi viete function:
5      while diff > epsilon
6          increment count by 1
7          sqrt_current = sqrt((2 + sqrt_previous)) (sqrt_previous's initial value is 0)
8          total ×= (sqrt_current / 2)
9
10         diff = absolute value of (sqrt_current - sqrt_previous)
11         sqrt_previous = sqrt_current (sqrt_current becomes the new sqrt_previous)
12
13     total = 2 / total
14     return total
15
16 define pi viete terms function:
17     return count
```

madhava.c : Madhava Series

$$p(n) = \sqrt{12} \sum_{k=0}^{n} \frac{(-3)^{-k}}{2k+1}$$

```
1  (this function makes use of the sqrt function provided in newton.c, which will be
2  referenced as sqrt() in this pseudocode)
3
4  define pi madhava function:
5      for (k = 0, absolute value of current_term > epsilon, increment k by 1)
6          increment count by 1
7          exp_fraction = 1 / exponent (exp refers to exponent)
8          bottom = (2 × k) + 1
9          current_term = exp_fraction / bottom
10         total += current_term
11         exponent ×= -3
12     total = sqrt(12) × total
13     return total
14
15 define madhava terms function:
16     return count
```

mathlib-test : Test harness for the previous functions.

```
define the options of the test harness "aebmrvnsh"
start main function:
    declare booleans for sqrt, e, and various pi function computations
    declare booleans for statistic printing, and program synopsis message
    while loop (use opt = getopt ( ... ) to take arguments out of the available options)
        switch case for (opt)
            case 'a'
                if triggered, set all booleans for cases "erbmvn" to true
            case 'e'
                if triggered, set boolean for e test to true
            case 'r'
                if triggered, set boolean for Euler test to true
            case 'b'
                if triggered, set boolean for BBP test to true
            case 'm'
                if triggered, set boolean for Madhava test to true
            case 'v'
                if triggered, set boolean for Viete test to true
            case 'n'
                if triggered, set boolean for Newton Sqrt tests to true
            case 's'
                if triggered, set boolean for statistic printing for cases "erbmvn" to true
            case 'h'
                if triggered, set boolean for synopsis message to true
            default
                if triggered, set boolean for synopsis message to true

(below is a list of conditionals that will print the corresponding tests to the cases
triggered above)
    define boolean that checks if any of the following tests (e, r, b, m, v, n) are being run

    if no tests are being run or the boolean in case 'h' or default is set off
        print the synopsis message

    (the rest of the conditionals will be checking for cases "erbmvn" individually with
    a nested if statement inside each of those individual cases that will be triggered if
    case 's' is triggered for printing statistics)

    if boolean for case . . . is true
        print corresponding results (mimicked <math.h> result, actual <math.h> result,
        and the difference between the two results)
        if boolean for case 's' is true
            print corresponding statistics

    (rest of the conditionals) . . .
    terminate program after all conditionals have been checked
```

4 Function Usage

1. Mathematical computation functions for e, sqrt, and pi:
    - These functions included inside e.c, newton.c, euler.c, bbp.c, viete.c, and madhava.c all take no parameters, except for the exception for the sqrt_newton() function inside of newton.c.
    - sqrt_newton() will take a **double** type value and produced the square root of said value. The sqrt_newton() function may also be included in the various files that include functions to compute pi (euler.c, bbp.c, viete.c, and madhava.c).
    - All of the mathematical computation functions should have no main() function defined inside of the file.
2. Test Harness (mathlib-test.c):
    - The test harness is the only ".c" file that should have a main function defined since all the messages being printed will be done so via this file.
    - Depending on what the user enters in the format of "-s", "-as", or "-a -s", etc. will trigger the switch case(s) to then trigger booleans to print the corresponding messages associated with the chosen option.
    - If the option entered is invalid, or no option is entered at all the program will simply print out the synopsis page explaining proper use of the program and the valid options available.

5 Variables

1. count
    - Each of the mathematical computation files (e.c, newton.c, euler.c, bbp.c, viete.c, and madhava.c) will have a static variable count. This will be used to keep track of how many iterations each of the functions take to compute a certain value.
2. total/result/upper
    - Most of the mathematical computations files (specifically e.c, euler.c, bbp.c, viete.c, and madhava.c) include a **double** type variable total or result which is where the approximated value e or pi which will be returned after the loop condition is not met.
    - newton.c on the other hand returns an upper bound which is approximated after each iteration until the loop condition is not met.

6 Credit

1. I attended Brian Mak's tutoring session on 10/4/21 for assistance to format my Makefile properly. Where I was guided to **not** produce binaries of any of the

mathematical computation ".c" files, and that there should only be one main() function defined inside of the test harness for the LDFLAGS to allow proper linkage.