Viveka Agrawal
Professor Veenstra
CSE13S, Winter 2023

CSE13S ASGN2 DESIGN DOC

Description of the program:

In this assignment, we are given different equations which must be coded in C. They include:

newton.c
Has 2 functions, sqrt_newton() which approximates the square root of an argument passed using the Newton-Raphson method. The other function is e_terms() which will return the number of iterations taken from sqrt_newton().

e.c

Has 2 functions, e() which approximates the value of e from the Taylor series and keeps count of the number of computed terms using a static variable which is local to the file. The other function is e_terms() which will return the number of computed terms.

madhava.c

Has 2 functions, pi_madhava() which approximates pi using the Madhava series and keeps count of the number of computed terms using a static variable which is local to the file. The other function is pi_madhava_terms() which will return the number of computed terms.

euler.c

Has 2 functions, pi_euler() which approximates pi using the formula derived from Euler's solution to the Basel problem, and keeps count of the number of computed terms using a static variable which is local to the file. The other function is pi_euler_terms() which will return the number of computed terms.

bbp.c

Has 2 functions, pi_bbp() which approximates the value of pi using the Bailey-Borwein-Plouffe formula, and keeps count of the number of computed terms using a static variable which is local to the file. The other function is pi_bbp_terms() which will return the number of computed terms.

viete.c

Has 2 functions, pi_viete() which approximates pi using Viete's formula and keeps count of the number of computed terms using a static variable which is local to the file. The other function is pi_viete_factors() which will return the number of computed terms.

newton.c

// The pseudocode for this function is provided by Professor Long

Include the standard library header, the boolean data type header, and mathlib.h which is the header file provided by Professor Long

Create an integer variable for the counter and set it equal to 0

Create the function for calculating square root and have the parameter as x

    create an integer variable and set that equal to 0.0 (z)
    create another integer variable and set that equal to 1.0 (y)
    reset the counter variable back to 0
    while the absolute value of (y - z) is greater than epsilon,
        set z equal to y
        set y equal to 0.5 times (z and add that to (x divided by z))
        increment the counter variable by 1
    return the value of y

Create the function for returning the counter number from the square root function

    Return the value of the counter variable value

e.c

Include the standard library header, the boolean data type header, and mathlib.h which is the header file provided by Professor Long

Create an integer variable for the counter and set it equal to 0

Create the function for calculating the value of e with a void parameter

    create an integer variable called denominator and set it equal to 1 // denominator of
        equation
    create an integer variable called k and set it equal to 0 // this is the index
    Create a double variable called i that calculates the current value of e

While true

Increment the index variable (k) by 1

Set the denominator variable to the denominator times the index variable (k)

// calculating denominator with new index

Set the current calculated value of e (i) to 1 divided by the denminator variable plus the current calculated value of e (i)

// calculates next term and add it to the current value of e

Increment the counter variable by 1

If 1 divided by the denominator is less than epsilon

Create a new double variable j and set it equal to 1 plus the current calculated value (i) // adds 1 to the current calculated value of e

Return the value of j

Break out of the while loop

Create the function for returning the counter number from the e function

Return the counter variable value plus 1 to get the same number of terms as Professor Long

madhava.c

Include the standard library header, the boolean data type header, and mathlib.h which is the header file provided by Professor Long

Create an integer variable for the counter and set it equal to 0

Create a function that sets the power of the first variable (a) to the second variable (b) // 1st and 2nd variables are passed as parameters

Create a double variable and set it equal to 1 (c)

If the 2nd passed variable is less than 0

Set the 2nd passed variable equal to -1 times the 2nd passed variable

Set the first passed variable equal to 1 divided by the 1st passed variable

Create an integer variable and set it equal to 1 (d)

While d is less than or equal to b

Set c equal to the value of c times a

Increment d by 1

Return the value of c

Create the function for calculating the madhava approximation of pi

create a double variable and set that equal to 0 (denominator)

create a double variable and set that equal to 1 (numerator)

create a double variable called k and set it equal to 0 // this is the index
Create a double variable called i that calculates the current value
While true
Set the numerator variable equal to the power function of -3 and -k // calculates the numerator for new index
Set the denominator equal to 2 times the index (k) plus 1 // calculates the denominator for new index
Set the value of i equal to the numerator divided by the denominator plus i // calculates the next term and adds it to previous value
Create a new double variable (p) and set it equal to the numerator divided by the denominator
Increment the index variable by 1
Increment the counter variable by 1
If p is less than 0
    Set p equal to the numerator divided by the denominator and multiply that by -1
    If p is less than epsilon
        Create a new double variable (j) and set it equal to the square root of 12 times i // calculates final value
        Return the value of j
        Break out the of the while loop

Create the function for returning the counter number from the madhava function
    Return the counter variable value minus 1 to get the same number of terms as Professor Long

euler.c

Include the standard library header, the boolean data type header, and mathlib.h which is the header file provided by Professor Long

Create an integer variable for the counter and set it equal to 0

Create the function for calculating the euler approximation of pi
Create a double variable and set it equal to 0 (y) // next term
create a double variable and set that equal to 2 (k) // index
create a double variable and set that equal to 1 (total) // current total calculated value
Create a double variable (squared)
While true
    Set squared equal to k times k // the denominator is squared
    Set y equal to 1 divided by squared

Set total equal to total plus y // adds next term value to current total
Increment index by 1
Increment counter by 1
If y is less than epsilon
Break out of the while loop
Set the total equal to total times 6 // multiply current total by 6 based on formula
Create a new double variable final and set it equal to the square root of the total
Return the value of final

Create the function for returning the counter number from the euler function
Return the counter variable value

bbp.c

Include the standard library header, the boolean data type header, and mathlib.h which is the header file provided by Professor Long

Create an integer variable for the counter and set it equal to 0

Create a function that sets the power of the first variable (a) to the second variable (b) // 1st and 2nd variables are passed as parameters
Create a double variable and set it equal to 1 (c)
If the 2nd passed variable is less than 0
Set the 2nd passed variable equal to -1 times the 2nd passed variable
Set the first passed variable equal to 1 divided by the 1st passed variable
Create an integer variable and set it equal to 1 (d)
While d is less than or equal to b
Set c equal to the value of c times a
Increment d by 1
Return the value of c

Create the function for calculating the bbp approximation of pi
Create a double variable and set it equal to 0 (denominator)
Create a double variable and set it equal to 0 (numerator)
Create a double variable and set it equal to 0 (k) // index
Create a double variable and set it equal to 0 (i) // current calculated value
While true
Set numerator equal to the power function of 16 and -k times (k times (120 times k plus 151) plus 47)

Set denominator equal to k times (k times (k times ((512 times k) plus 1024) plus 712) plus 194) plus 15
Set i equal to the numerator divided by the denominator plus i // calculate next term and add to current value
Create a double variable (p) and set it equal to numerator divided by denominator
Increment index by 1
Increment counter by 1
If p is less than epsilon
Return the value of i
Break out of the while loop

Create the function for returning the counter number from the bbp function
Return the counter variable value

viete.c

Include the standard library header, the boolean data type header, and mathlib.h which is the header file provided by Professor Long

Create an integer variable for the counter and set it equal to 0

Create the function for calculating the viete approximation of pi
Create a double variable and set it equal to 0 (numerator)
Create a double variable and set it equal to 1 (k) // index
While true
Set the numerator equal to the square root of 2 plus the numerator
Set k equal to the (numerator divided by 2) times k // calculate next term and multiplying it by current value
Create a double variable (denominator) and set it equal to the numerator divided by 2
Increment counter by 1
If 1 divided by the denominator minus 1 is less than epsilon
Create a double variable (a) and set it equal to 2 divided by k // calculate final value
Return the value of a
Break out of the while loop

Create the function for returning the counter number from the viete function
Return the counter variable value minus 1 to get the same number of terms as Professor Long

mathlib-test.c

Include the standard library header, the boolean data type header, the input/output header, the string header, the header that allows multiple decimal places to be printed, the math library, and mathlib.h which is the header file provided by Professor Long

Define OPTIONS “aebmrvnsh”
Create the main function with the parameters argc and char **argv
    Create an integer variable (opt) and set it equal to 0
    Create a boolean variable (no_user_input) and set it equal to true
    Create a character array of size 10 // for storing command line arguments when parsing
    Create an integer variable (i) and set it equal to 0 // index variable
    Create a double variable (calcvalue) and set it equal to 0 // variable to store calculated value

    Create boolean variable (test_a) and set it to false
    Create boolean variable (test_b) and set it to false
    Create boolean variable (test_m) and set it to false
    Create boolean variable (test_r) and set it to false
    Create boolean variable (test_v) and set it to false
    Create boolean variable (test_n) and set it to false
    Create boolean variable (test_s) and set it to false
    Create boolean variable (test_h) and set it to false

    // This while loop will read and parse through the command line to check for repeated command line options and if -s is present anywhere in the command line
    while opt equals getopt with the arguments argc, argv, OPTIONS and it is not equal to 1
        Set no_user_input to false
        Use a switch statement with the parameter opt
            For case a
                If not test_a
                    Set the index i of the array equal to opt // this stores the nonrepeated argument in the array
                    Increment i
                    Set test_a equal to true
                    break
            For case e
                If not test_e
                    Set the index i of the array equal to opt // this stores the nonrepeated argument in the array

```
                    Increment i
                    Set test_e equal to true
                    break
    For case b
            If not test_b
                    Set the index i of the array equal to opt // this stores the
                            nonrepeated argument in the array
                    Increment i
                    Set test_b equal to true
                    break
    For case m
            If not test_m
                    Set the index i of the array equal to opt // this stores the
                            nonrepeated argument in the array
                    Increment i
                    Set test_m equal to true
                    break
    For case r
            If not test_r
                    Set the index i of the array equal to opt // this stores the
                            nonrepeated argument in the array
                    Increment i
                    Set test_r equal to true
                    break
    For case v
            If not test_v
                    Set the index i of the array equal to opt // this stores the
                            nonrepeated argument in the array
                    Increment i
                    Set test_v equal to true
                    break
    For case n
            If not test_n
                    Set the index i of the array equal to opt // this stores the
                            nonrepeated argument in the array
                    Increment i
                    Set test_n equal to true
                    break
    For case s
            If not test_s
```

```
                Set the index i of the array equal to opt // this stores the
                    nonrepeated argument in the array
                Increment i
                Set test_s equal to true
                break
        For case h
            If not test_h
                Set the index i of the array equal to opt // this stores the
                    nonrepeated argument in the array
                Increment i
                Set test_h equal to true
                break
        otherwise default break

// This for loop processes nonrepeated command line arguments
for integer j equals 0, j is less than i, and increment j
    Set opt equal to the array with index j
    Use a switch statement with the parameter opt
        For case a
            Set calcvalue equal to the e function
            Print the approximation of e, the value of M_E, and the difference
                between the 2
            If test_s is true
                Print the number of e terms
            Set calcvalue equal to the euler function
            Print the approximation of euler, the value of M_PI, and the
                difference between the 2
            If test_s is true
                Print the number of euler terms
            Set calcvalue equal to the bbp function
            Print the approximation of bbp, the value of M_PI, and the
                difference between the 2
            If test_s is true
                Print the number of bbp terms
            Set calcvalue equal to the madhava function
            Print the approximation of madhava, the value of M_PI, and the
                difference between the 2
            If test_s is true
                Print the number of madhava terms
            Set calcvalue equal to the viete function
```

Print the approximation of viete, the value of M_PI, and the difference between the 2
If test_s is true
Print the number of viete terms
Set calcvalue equal to the square root function
For double integer i equal 0.000000000000000, i less than 10.000000000000000, i equals i plus 0.100000000000000
Set calcvalue equal to the square root function of i
Print the approximation of square root, the actual value of square root, and the difference between the 2
If test_s is true
Print the number of square root terms
Break out of case a
For case e
Set calcvalue equal to the e function
Print the approximation of e, the value of M_E, and the difference between the 2
If test_s is true
Print the number of e terms
Break out of case e
For case b
Set calcvalue equal to the bbp function
Print the approximation of bbp, the value of M_PI, and the difference between the 2
If test_s is true
Print the number of bbp terms
Break out of case b
For case m
Set calcvalue equal to the madhava function
Print the approximation of madhava, the value of M_PI, and the difference between the 2
If test_s is true
Print the number of madhava terms
Break out of case m
For case r
Set calcvalue equal to the euler function
Print the approximation of euler, the value of M_PI, and the difference between the 2
If test_s is true
Print the number of euler terms

Break out of case r
For case v
Set calcvalue equal to the viete function
Print the approximation of viete, the value of M_PI, and the difference between the 2
If test_s is true
Print the number of viete terms
Break out of case v
For case n
Set calcvalue equal to the square root function
For double integer i equal 0.000000000000000, i less than 10.000000000000000, i equals i plus 0.100000000000000
Set calcvalue equal to the square root function of i
Print the approximation of square root, the actual value of square root, and the difference between the 2
If test_s is true
Print the number of square root terms
Break out of case n
For case s
Set test_s equal to true
Break out of case s
For case h
Print the help message
Break out of case h
Otherwise default break
If no_user_input is true
Print the help message
Return 0

mathlib.h

The main test harness for the implemented math library which will be compared against math.h. Use getopt() to parse.

Has the command line options:

-a: Runs all tests.

-e: Runs e approximation test.

-b: Runs Bailey-Borwein-Plouffe π approximation test. -m: Runs Madhava π approximation test.

-r: Runs Euler sequence π approximation test.

-v: Runs Viète π approximation test.

-n: Runs Newton-Raphson square root approximation tests.

-s: Enable printing of statistics to see computed terms and factors for each tested function.

-h: Display a help message detailing program usage.

Create a Makefile to compile the code.