Assignment 2 DESIGN.pdf

Ning Jiang

Description of Program:

mathlib.c

Implement a small library of mathematical functions, including Exponent function, Sine function, Cosine function, Square root function, log function and integrate function.

Files to be included in directory “asgn2”:

functions.c: This file is provided and contains the implementation of the functions that your main program should integrate.

2. functions.h: This file is provided and contains the function prototypes of the functions that your main program should integrate.

3. integrate.c: This contains and the main() function to perform the integration specified by the command-line over the specified interval.

4. mathlib.c: This contains the implementation of each of your math library functions.

mathlib.h: This file is provided and contains the interface for your math library.

README.md: This must use proper Markdown syntax. It must describe how to use your script and Makefile. should also list and explain any command-line options that your program accepts.

DESIGN.pdf: This document must be a proper PDF. Describe my design and design process for your program with enough detail. Describe how your program works with supporting pseudocode.

WRITEUP.pdf: This document must be a proper PDF. This writeup must include the plots that you produced using your bash script, as well as discussion on which UNIX commands you used to produce each plot and why you chose to use them.

Pseudocode / Structure:

From asgn2.pdf

Implementation of e^x

```
def exp(x, epsilon = 1e-14):
    trm = 1.0
    sum = trm
    k = 1
    while trm > epsilon:
        trm *= abs(x) / k
        sum += trm
        k += 1
    return sum if x > 0 else 1 / sum
```

Implementation of $\sin(x)$

```
def sin(x, epsilon = 1e-14):
    s, v, t, k = 1.0, x, x, 3.0
    while abs(t) > epsilon:
        t = t * (x * x) / ((k - 1) * k)
        s = -s
        v += s * t
        k += 2.0
    return v
```

Cos(x)

```
def sin(x, epsilon = 1e-14):
  s, v, t, k = 1.0, 1.0, 1.0, 2.0
  while abs(t) > epsilon:
      t = t * (x * x) / ((k - 1) * k)
      s = -s
      v += s * t
      k += 2.0
  return v
```

Implementation of $\sqrt{x}$

```
def sqrt(x, epsilon = 1e-14):
    z = 0.0
    y = 1.0
    while abs(y - z) > epsilon:
        z = y
        y = 0.5 * (z + x / z)
    return y
```

Implementation of $\log(x)$

```
def log(x, epsilon = 1e-14):
    y = 1.0
    p = exp(y)
    while abs(p - x) > epsilon:
        y = y + x / p - 1
        p = exp(y)
    return y
```

Simpson's rule

```
def simpson_38(f, a, b, n):
  h = (b - a) / n
  sum = f(a) + f(b)
  for i in range(1, n):
      if i % 3 != 0:
```

```
6         sum += 3 * f(a + i * h)
7   else:
8      sum += 2 * f(a + i * h)
9   sum *= h * 3 / 8
10  return sum
```

Integrate.c

```
# define OPTION "abcdefghijHn:p:q:"
void usage(char *exec):
   This Function will just printout the program’ s usage and
synopsis.

def main(argc, argv):
    int a_integrate = 0;
    int b_integrate = 0;
    int c_integrate = 0;
    int d_integrate = 0;
    int e_integrate = 0;
    int f_integrate = 0;
    int g_integrate = 0;
    int h_integrate = 0;
    int i_integrate = 0;
    int j_integrate = 0;
    While ((option = getopt(argc, argv, OPTION)) != -1):
       switch (option)
        If case is p:
            low = input(double)
        If case is q:
            High = input(double)
        If case is n:
           Partition = input(int)
        If case is a:
            a_integrate = 1;
            break;
        If case is b:
            b_integrate = 1;
            break;
        If case is c:
            c_integrate = 1;
            break;
        If case is d:
            d_integrate = 1;
            break;
        If case is e:
```

```
            e_integrate = 1;
            break;
        If case is f:
            f_integrate = 1;
            break;
        If case is g:
            g_integrate = 1;
            break;
        If case is h:
            h_integrate = 1;
            break;
        If case is i:
            i_integrate = 1;
            break;
        If case is j:
            j_integrate = 1;
            break;
        If case is H:
            Print usage(argv[0]);
    if (a_integrate = True):
            print(sqrt(1 - x^4), low, high, partition);
    else if (b_integrate = True):
            print(1/log(x), low, high, partition);
    else if (c_integrate = True):
            print(e^((−x)^2), low, high, partition);
    else if (d_integrate = True):
            print(sin(x^2), low, high, partition);
    else if (e_integrate = True):
            print("cos(x^2), low, high, partition);
    else if (f_integrate = True):
            print(log(log(x)), low, high, partition);
    else if (g_integrate = True):
            print("sin(x)/x, low, high, partition);
    else if (h_integrate = True):
            print((e^(−x))/x, low, high, partition);
    else if (i_integrate = True):
            print("e^(e^x), low, high, partition);
    else if (j_integrate = True):
            printf(sqrt((sin(x))^2+(cos(x))^2),    low,    high,
partition);
    # print the first line output like:
    # <function>,<low>,<high>,<partitions>

    for (uint32_t x = 2; x <= partition; x += 2) {
```

```
        if a_integrate is True:
            printf(x, integrate(%0.15a, low, high, x));
        else if b_integrate is True:
            printf(x, integrate(%0.15b, low, high, x));
        else if c_integrate is True:
            printf(x, integrate(%0.15c, low, high, x));
        else if d_integrate is True:
            printf(x, integrate(%0.15d, low, high, x))
        else if e_integrate is True:
            printf(x, integrate(%0.15e, low, high, x));
        else if f_integrate is True:
            printf(x, integrate(%0.15f, low, high, x));
        else if _integrate is True:
            printf(x, integrate(%0.15g, low, high, x));
        else if h_integrate is True:
            printf(x, integrate(%0.15h, low, high, x));
        else if i_integrate is True:
            printf(x, integrate(%0.15i, low, high, x));
        else if j_integrate is True:
            printf(x, integrate(%0.15j, low, high, x));
# print The subsequent lines looks like: <partition>,<value>
```

Notes:

1. How to include mathlib.c in integrate.c?

We can use # include “mathlib.h”

2. How to call another function?

We can use function pointer looks like: double (*f)(double)

And use & to all another function like: &a

3. Man getopt to see how to use getopt()

Error Handing:

1. I didn’t understand how to independently verify each important component, so I attend the tutoring section Tuesday with Miles.
2. Attended Wednesday tutoring section with Brian to ask about how to make Makefile.
3. Attended Wednesday tutoring section with Audrey to ask about how to ask how does main function works.
4. My else...if method in main function didn’t work, so I attended thuesday tutoring section with Miles to fix that problem.

Credit:

1. I learned how to use function pointer and getopt() which are really important.

2. I learned how to include another another file in a c file.

Simpson’s Rule: