Assignment 2 Design Document - Slice of PI

Description of the assignment:
We programmed our very own math library, which will be able to estimate the value pi and e. By implementing sqrt(x) and multiple math formulas/series that estimate, we can compare our work with the C math library and see how close we can get. We will make a test harness that will take in input to then print out our values and compare it with the value from the math library.

To go into specifics on how it works: We were provided with 5 different math formulas that approximate e or pi, all of which are in the form of infinite series. Sadly we cannot iterate over infinity for our approximation since it would take forever and we would not have a memory location large enough to hold this infinite approximation. Therefore, we stop the iterations when the current step is equal to or less than a predetermined EPSILON, in this case 1e-14. Specifically in my code this sends a break statement to the while loop to exit, and then returns the asked value. For our test harness, we use a function called getopt, where we can specify arguments in our run command to get specific outputs. The getopt will read what arguments are specified, and then use a switch case loop to do what the specific case for that argument is written to do. In my case, I do not do all of my printing in my cases, since it makes the code somewhat easier to read. In each case, a boolean variable is set to 1 to state that that specific input was seen. Out of the while loop of the getopt function, we can now print what we need to since we have exactly what the inputs are.

Implementations:
Newton.c: creates square root function
Bbp.c: creates a PI approximation using the bbp formula
Madhava.c: creates a PI approximation using the madhava formula
Euler.c: creates a PI approximation using the euler formula
Viete.c : creates a PI approximation using the viete formula
E.c : creates an e approximation using the e formula

Mathlib-test.c: creates a test harness that uses getopt to scan for input, in the form of letters, to specify which test you want displayed and whether you want statistics to be displayed or not.

Getopt inputs:

-a	Run all tests
-e	Runs e test
-b	Runs BBP pi test
-m	Runs Madhava pi test
-r	Runs Euler pi test

-v	Runs Viete pi test
-n	Runs Newton square root tests
-s	Print statistics
-h	Print help menu

How to Compile:
Since we have so many c files, it's imperative to first do:

```
$make clean
```

to make sure all of our old executables are removed. Then we:

```
$make
```

to compile the code.
How to Run:
To run the code, use this command:

```
$./mathlib-test
```

Pseudo Code:

E:

```
Sum = 1
K = 1
Fact = 1
while(1)
    Fact *= k
    if (1/fact <= epsilon)
        Break
    Else
        sum+=(1/fact)
        K++
Return sum
```

Madhava:

```
sum = 1
K = 1
Numerator = 1
while(1)
    Numerator *= -1/3
    Denominator = 2k + 1
    if(numerator/denominator <= epsilon)
        Break
    Else
        Sum += numerator/denominator
```

```
                K++
```

Euler:

```
Sum = 1
K = 1
while(1)
        if(1./k*k <= epsilon)
                Break
        Else
                Sum += 1./k*k
                K++
```

BBP:

```
Sum = 47/15
K = 1
K_power = 1
while(1)
        K_power = 1/16
        Int 8k = 8*k
        Secondpart
        if(k_power/second part <= epsilon)
                Break
        Else
                Sum += k_power/second part
                K++
                Print
```

Viete:

```
Sum = sqrt(2)/2
Numerator = sqrt(2)
K = 1
Epsilon = 1e-15
while(1)
        Numerator = sqrt(sqrt(numerator))
                If sqrt(sqrt(numerator)) <= epsilon
                        Break
                Else
                        Add to sum
```

Mathlib-test:

Use the getopt code provided to us in the assignment pdf
Switch case each input, and if seen, store in a specific boolean variable, such as “is_a” or “is_v”.
After the while loop in the getopt code, write if statements to check which inputs were imputed using our boolean variables.

In if statements, print returned value of specified approximator, with math library value, as specified by the binary file.
Functions were made to minimize the usage of the print statement, called printstatement() for the approximations and termprint() for the terms.