Formulas:

Euler’s Solution, the sum of the infinite series

$$\sum_{k=1}^{\infty} \frac{1}{k^2} = \frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} + \cdots = H_{\infty}^{(2)},$$

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

The Madhava Series:

[illegible]

$$\sum_{k=0}^{\infty} \frac{(-3)^{-k}}{2k+1} = \sqrt{3}\tan^{-1}\frac{1}{\sqrt{3}} = \frac{\pi}{\sqrt{12}}$$

but it gives us a more rapidly converging series, given by:

$$p(n) = \sqrt{12}\sum_{k=0}^{n} \frac{(-3)^{-k}}{2k+1} = \sqrt{12}\left[\frac{1}{2}3^{-n-1}\left((-1)^n \Phi\left(-\frac{1}{3}, 1, n+\frac{3}{2}\right) + \pi 3^{n+\frac{1}{2}}\right)\right].$$

Which leaves us with the problem of calculating $\sqrt{12}$. You may be thinking: “Can’t I just use the library?” The answer, of course, is *no*, you will need to compute $\sqrt{12}$ on your own. Do you need to worry about Φ? No, you just need to understand that in the limit, the Φ term (called the *Lerch transcendent*) goes to *zero* and that the remaining term

$$\frac{\pi}{2}3^{-n-1}3^{n+\frac{1}{2}} = \frac{\pi}{2\sqrt{3}} = \frac{\pi}{\sqrt{12}}.$$

The Wallis Series:

[illegible]

$$p(n) = 2\prod_{k=1}^{n} \frac{4k^2}{4k^2-1} = \frac{\pi\Gamma(n+1)^2}{\Gamma\left(n+\frac{1}{2}\right)\Gamma\left(n+\frac{3}{2}\right)}.$$

It is easy to calculate, but how rapidly does it converge? What is this Γ function? Do we need to compute it? How do we know this converges to π? Let’s factor out π, then take the limit and note that

$$\lim_{n\to\infty} \frac{\Gamma(1+n)^2}{\Gamma\left(\frac{1}{2}+n\right)\Gamma\left(\frac{3}{2}+n\right)} = 1.$$

The Bailey-Borwein-Plouffe Formula:

$$p(n) = \sum_{k=0}^{n} 16^{-k}\left(\frac{4}{8k+1} - \frac{2}{8k+4} - \frac{1}{8k+5} - \frac{1}{8k+6}\right).$$

And if you desire to reduce it to the least number of multiplications, you can rewrite it in *Horner normal form*:

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

Viete's Formula(not as accurate as others):

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

Or more simply,

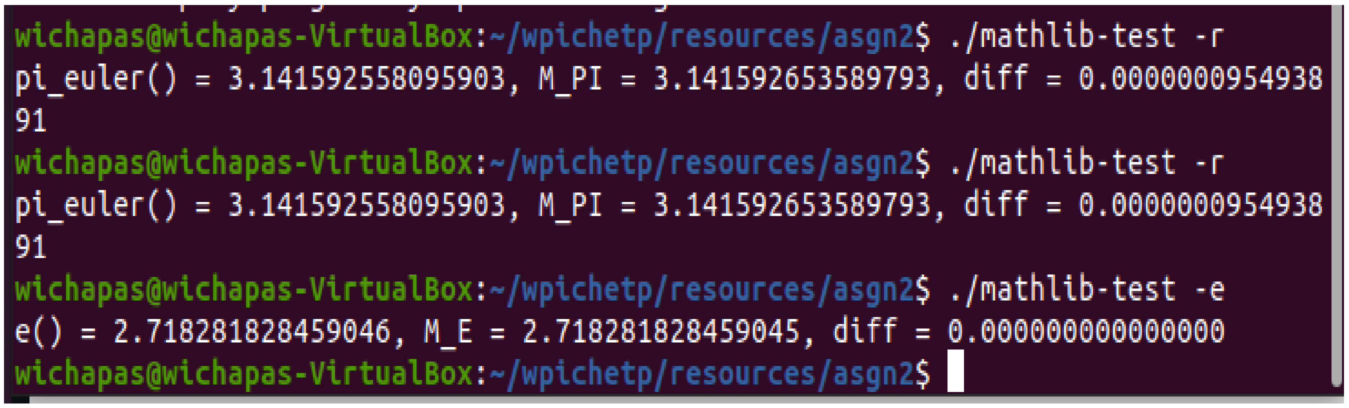
$$\frac{2}{\pi} = \prod_{k=1}^{\infty} \frac{a_i}{2}$$

where $a_1 = \sqrt{2}$ and $a_k = \sqrt{2 + a_{k-1}}$ for all $k > 1$.

Fastest Series:

$$\frac{1}{\pi} = \frac{2\sqrt{2}}{9801} \sum_{k=0}^{\infty} \frac{(4k)!(1101+26390k)}{(k!)^4 396^{4k}}$$

Expected input/output:



```
wichapas@wichapas-VirtualBox:~/wpichetp/resources/asgn2$ ./mathlib-test -r
pi_euler() = 3.141592558095903, M_PI = 3.141592653589793, diff = 0.0000000954938
91
wichapas@wichapas-VirtualBox:~/wpichetp/resources/asgn2$ ./mathlib-test -r
pi_euler() = 3.141592558095903, M_PI = 3.141592653589793, diff = 0.0000000954938
91
wichapas@wichapas-VirtualBox:~/wpichetp/resources/asgn2$ ./mathlib-test -e
e() = 2.718281828459046, M_E = 2.718281828459045, diff = 0.000000000000000
wichapas@wichapas-VirtualBox:~/wpichetp/resources/asgn2$ 
```

Looking from the test file, I know that we are comparing how well our formula does to the math library and we just subtract them to get the difference between them. It is possible to detect errors in our code if the difference is off from the test file. First thing to focus on is getting the formulas down. Next, to test if the code works I can implement getopt() to ask for certain

Rough Pseudocode (Not Python but Python style):

E.c:

```
Factorial_total = 0
K = 0
E function:

While loop (go on forever):

    If k = 0,
        factorial total = 1
        K += 1
        continue
    K += 1
    Factorial_total = factorial_total * k
    E_formula = 1 /factorial_total
    If e_formula < EPSILON:
        Break
    E_total += e_formula
    E_term_tracker[0] += 1
Return e_total
```

Only need a while loop to make this summation. Have a k counter always as it always increased by 1 and a factorial counter to keep track of your factorial calculations. The factorial calculations will stack up for later summations. You will want to store what you calculated into e_total so you can return the total sum later.

Euler.c:

```
Int k = 1
Euler sum = 0

While euler_pow = k * k
Euler_formula = 1/euler_pow
If euler_formula < EPSILON, break
Euler_sum += euler_formula
k = k + 1

Return euler_sum
```

This program should not take in any input but just loop using a while loop because we're making a summation, similarly to the follow formulas.

Newton.c:

```
Y = 1.0

Assert if x >= 0

For guess =0, absolute (y - guess) > EPSILON; y = (y+x/y) / 2
        Guess = y
Return y
```

Calculates and returns a sqrt value

Bbp.c:

```
While true…
        If (k ==0)
                bbp _formula = replicate the formula
                Bbp_total += bbp_formula
                Bbp_pow += 1
                K += 1
        Bbp_formula = replicate the formula
        Bbp += bbp_formula
```

```
    Bbp_pow *= 16
```

Just follow the formula but remember the power just increases by multiplying 16 each time

Madhava.c:

```
While true…

    If k == 0

        Madhava_pow += 1
        Madhava_pow = madhava_pow / (2*k+1)
        Madhava_total += madhava_formula
        K += 1
```

All of the formulas can be easily replicated but keeping track of the counter is the big thing and working with types in this lab.