HOMEWORK 2

Due Thursday, 26 Jan, at 11pm

Please enter your answers into a Jupyter notebook and submit by the deadline via canvas.

Problem 1. Playing computer. Each of the following functions is computing a mathematical function. Describe, in english or math, what each function is computing. To fiure it out, you will probably need to choose input values (a.k.a. arguments) in your head and simulate the running of the program in your head or on paper. Of course, you could just code them in and plug in examples, but that would defeat the purpose of this whole problem. Write down your answers in markdown in your Homework02.ipynb.

```
def c(n):
    return True if n % 2 == 0 else False
```

```
def ff(n):
    answer = 1
    while n>1:
        answer = answer * n
        n = n - 1
    return answer
```

```
def f(n):
    answer = 0
    while n > 1:
        n = n//2
        answer = answer + 1
    return answer
```

```
def g(x):
    y = x
    while y >= 1:
        y = y - 1.0
    return x - y
```

```
def h(n):
    i = 2
    while i<n:
        if n % (i*i) == 0:
            return True
        i += 1
    return False
```

Optional: The above functions don't always work (e.g. g won't do something useful if $x < 0$, for each function, figure out which values of the arguments will be ok)

Problem 2. This is a little exercise that supposedly people used to get asked about as a warm-up in programming interviews. I am not sure if this is true, but the lore remains. If you go to a programming interview, and they asked you some basic programming exercises you would do in HW2 of your first programming class, you can say they asked you "FizzBuzz" type questions:

Write a program that prints the numbers from 1 to 100. But for multiples of three print Fizz instead of the number and for the multiples of five print Buzz. For numbers which are multiples of both three and five print FizzBuzz.

Problem 3. The prime number theorem. The prime number theorem states that as n becomes large, the number of prime numbers between 1 and n approaches $\frac{n}{\log n}$ where $\log = \log_e$ is the natural logarithm. Write a function `num_primes(n)`, that will give you the number of primes between 1 and n.

Then, for $n = 10, 100, 1000, \ldots, 10^6$, print out $n/\log n$ and `num_primes(n)`.

Problem 4. Write a function `factors(n)` that will print all the divisors of a number n. Then write a function `prime_factors(n)` that will print only the prime factors of a number n.

Problem 5. Perfect numbers. In number theory, a perfect number is a positive integer that is equal to the sum of its proper positive divisors, that is, the sum of its positive divisors excluding the number itself. Example : The first perfect number is 6, because 1, 2, and 3 are its proper positive divisors, and $1 + 2 + 3 = 6$. The next perfect number is $28 = 1 + 2 + 4 + 7 + 14$.

Write a loop that will find next two perfect numbers.

The perfect number after those two is much larger (8 digits) and would take a very long time to get. Optional: by timing your existing program, estimate how long it would take your program to to check all 8 digit numbers.