COM1310 Assignment # 5 (2022)

Each of these questions should be answered using complete sentences/paragraphs, plus supporting diagrams, tables, and formulas where appropriate. Don't just tell what you know; tell how and why you know it!

1. Use induction to prove that $6|(7^n - 1)$ for all $n \geq 0$.

2. Let F_n = nth Fibonacci number

   Recall that the Fibonacci sequence satisfies the recurrence formula $F_{n+1} = F_n + F_{n-1}$ with $F_1 = F_2 = 1$.

   Prove using induction that for all n > 1, F_{n+1} and F_n are relatively prime. (Partial credit for producing a WOP proof instead of using induction)

3. a) Prove that $gcd(n, n+1) = 1$ for any positive integer n, and use that to justify a conclusion that if p is prime then $p \mid n$ implies p does not divide $n + 1$.

   b) Prove that there are infinitely many primes.

   Hint: Assume $p_1, p_2, \ldots, p_k$ are all the primes and consider $n = p_1 p_2 \ldots p_k + 1$.

4. a) Compute $5^{1024}\ (mod\ 28)$ using the Euclidean algorithm and Euler's Theorem.

   Hint: You might use $\Phi(ab) = \Phi(a)\,\Phi(b)$ if a and b are relatively prime.

   b) Use part a) to compute $2^{5^{1024}}\ (mod\ 29)$.

   Hint: 29 is prime.

5. **Calculating Prime Factorizations and Euler's Totient Efficiently**. One of the neat things about Python its extensibility. Suppose you need something that can work in a loop like range(N), but instead will only generate all the *prime* numbers up to N. This is very simple to do.

   - First, write a Python function that will *print* all the primes up to some limit N
   - Then change the word *print* to *yield*, and you are done.

   Presto! The function has become a *generator* that returns a lazy sequence of primes back to its caller instead of printing them out. You can use it like a range object. Much more versatile!

   A century after Philo began to imply stuff and Euclid started pulverizing, a renowned polymath names Eratosthenes (mathematician, astronomer, geometer, poet, literary critic, philosopher, geographer, and head librarian of the largest library in the world) devised a wickedly clever way to enumerate all the primes up to some limit with near-optimal efficiency. The *Sieve of Eratosthenes* has been provided to you as a Python generator in a separate file for use in this problem. Import it and call it. For example:

```
import sieve from sieve
for prime_number in sieve(20):
    print(prime_number)
```

will print the primes from 2 up through 20. You can learn more about how the algorithm works at https://en.wikipedia.org/wiki/Sieve_of_Eratosthenes and you can learn more about its inventor here: https://en.wikipedia.org/wiki/Eratosthenes. For this assignment, you can use it as a black box.

For this assignment you will write a Python file called `totient.py` that contains two functions:

- The function `prime_fac` accepts an integer parameter `n` and returns the prime factorization of `n` as a list of factors and their powers. For example, the prime factorization of 24 is $2^3 3^1$ so `prime_fac(24)` would return the list `[(2,3),(3,1)]`. The factors are in ascending order. This function must use the sieve code provided as its generator of prime numbers.
- The function phi accepts a prime factorization list and returns the Euler totient function for that factorization (e.g., 8 for the factorization listed above).

The file `totient.py` shall contain only these two functions, plus the import statement for `sieve.` Any test code you write should be in another file (that you don't turn in) that imports from `totient.py`.

Upload only `totient.py` to the code dropbox for this assignment.