ECE 406, Fall 2020, Assignment 1

General rule: whenever a problem asks you, "what is..." or "true or not necessarily true...," you must *always* provide a justification. Your justification must not be some elaborate prose. Rather, it should be punchy, to-the-point, technical and precise.

Marks: all problems are worth the same, with equal distribution over sub-problems. The assignment is worth 50/4 towards your mark in the course. As the "syllabus, logistics and schedule" document says, we may mark a subset of these problems only.

1. Your textbook says in Section 0.1, "[Using] the decimal notation...even very large numbers could be written down compactly..." It says this specifically in comparison to the Roman numeral system. In this problem, we assess the veracity of this claim.

   Adopt the following for what the Roman numeral system is to encode a positive integer (credit: `https://www.britannica.com/topic/Roman-numeral`):

   > The symbols are I, V, X, L, C, D, and M, standing respectively for 1, 5, 10, 50, 100, 500, and 1,000 in the Hindu-Arabic numeral system. A symbol placed after another of equal or greater value adds its value; e.g., II = 2 and LX = 60. A symbol placed before one of greater value subtracts its value; e.g., IV = 4, XL = 40, and CD = 400. A bar placed over a number multiplies its value by 1,000. A bar is not allowed over another bar.

   An example of the use of a bar is $\overline{\text{VII}}\,\text{I}$, which represents 7,001. Of course, another way to represent 7,001 is MMMMMMMI. As a bar is not allowed over another bar, to represent 7,000,000 for example, $\overline{\overline{\text{VII}}}$ is *not* legal.

   For the following parts (a) and (b), assume we adopt the most compact encoding for the Roman numerals. So, for example, $7,001$ would be written as $\overline{\text{VII}}\,\text{I}$ (5 symbols only, counting the bar as one) and not MMMMMMMI (8 symbols).

   (a) What, in $\Theta(\cdot)$ notation, is the worst-case number of symbols we need to write down to encode a positive integer n in the Roman numeral system? Treat a bar up top as one symbol, immaterial of the length of the bar.

   (b) To compare your solution to (a) to the worst-case in the decimal (base-10) system for a positive integer, identify, in $\Theta(\cdot)$ notation, the worst-case number of symbols to encode a positive integer in decimal notation. Once you do so, state tersely whether the decimal encoding is the same, better or worse than (a).

2. The series $1/1 + 1/2 + \ldots + 1/n$ shows up later in the course.

   Prove: $\displaystyle\sum_{i=1}^{n} \frac{1}{i} = O(\log n)$.

   Hint: adopt the exact form $\log_b n + c$ *for the right hand side. Decide what choices for the constants* b, c *would be prudent towards carrying the proof through. Then prove by induction on* n*.*

3. Consider the following recurrence for $\gcd(a, b)$ for $a, b \in \mathbb{Z}^+$, $a \geq b$:

$$\gcd(a,b) = \begin{cases} a & \text{if } b = 0 \\ 2 \cdot \gcd(a/2, b/2) & \text{if both } a, b \text{ are even} \\ \gcd(a, b/2) & \text{if } a \text{ is odd and } b \text{ is even} \\ \gcd(b, a) & \text{if } a \text{ is even and } b \text{ is odd} \\ \gcd((a-b)/2, b) & \text{otherwise} \end{cases}$$

   (a) Prove that the recurrence is correct.

   (b) What is the worst-case time-efficiency in $O(\cdot)$ as a function of the size n of the input $\langle a, b \rangle$ of a recursive algorithm that realizes the recurrence?

4. We want an algorithm to compute $n!$ for some $n \in \mathbb{N} = \{1, 2, \ldots\}$. Consider the following two candidates, where FACTTWO is initially invoked as FACTTWO$(1, n)$.

```
FACTONE(n)
1 if n = 1 then return 1
2 return n × FACTONE(n − 1)
```

```
FACTTWO(lo, hi)
11 if lo = hi then return lo
12 mid ← ⌊(lo + hi)/2⌋
13 return
       FACTTWO(lo, mid) × FACTTWO(mid+1, hi)
```

   (a) State a meaningful correctness property for FACTTWO on input $\langle lo, hi \rangle$ where $lo, hi \in \mathbb{N}$, $lo \leq hi$, and prove that the algorithm indeed possesses that property.
   Hint: induction on $hi - lo + 1$.

   (b) Adopt the total number of executions of Line (2) as the measure for running-time of an invocation FACTONE(n). State in $\Theta(\cdot)$ notation what that is as a function of n. Similarly, adopt the total number of executions of Line (13) as the measure for running-time of an invocation FACTTWO$(1, n)$. State in $\Theta(\cdot)$ notation what that is as a function of n. E.g., we observe that if we invoke FACTONE(3), the total number of times Line (2) is executed is 2.

5. **[python3]** Devise and implement a polynomial-time algorithm to compute $x^{y^z} \bmod p$ given input $\langle x, y, z, p \rangle$ where $x, y, z, p \in \mathbb{N} = \{1, 2, 3, \ldots\}$ and p is prime.

   "Polynomial-time" in this context of course means: in time polynomial in $\log(\max\{x, y, z, p\})$.

   Hint: exploit Fermat's little theorem.

   Submission: submit two things:

   (a) On Learn, there is a skeleton `a1p5.py` file. You should populate that with your code and upload to the dropbox on Learn. Your submission must be named `a1p5.py` . There is also a tester file which has a few test-cases. However, the TAs will likely add their own test-cases when they mark. So you should test your code more comprehensively than with the provided tests. Better still, you should stare at your code, i.e., engage in code-inspection, to gain confidence that it is correct.

   (b) A blank page here on Crowdmark. We will provide you your marks on Crowdmark.