1.2 Let $l \in \mathbb{N}$. Note that the number of bits needed to represent l is $[\ln(l)/\ln(2)] + 1$ and the number of decimal places needed to represent l is $[\ln(l)/\ln(10)] + 1$ where [] is the floor function. Therefore,

$$\begin{aligned}
\frac{[\ln(l)/\ln(2)] + 1}{[\ln(l)/\ln(10)] + 1} &= \frac{[\ln(l)/\ln(2)] + 1}{[\ln(l)/\ln(10)] + 1} \frac{\frac{1}{\ln(l)}}{\frac{1}{\ln(l)}} \\
&\leq \frac{\frac{1}{\ln(l)}[\ln(l)/\ln(2)] + \frac{1}{\ln(l)}}{\frac{1}{\ln(10)}} \\
&= \frac{\log(10)}{\ln(l)}[\ln(l)/\ln(2)] + \frac{\log(10)}{\ln(l)} \\
&\leq \frac{\ln(10) + 1}{\ln(2)}
\end{aligned}$$

the ratio is approximately $\ln(10)/\ln(2)$. Note that since $16 > 10 > 8$, then the ratio is bounded above by $\ln(16)/\ln(2) = 4(\ln(2/\ln(2)) = 4$. In the limit, the ratio should approximate the quantity $\ln(10)/\ln(2)$.

1.3 The i-th layer in the tree can store d^{i-1} by itself. Therefore if n, where n is the number of nodes in this rooted tree, is equal to $1 + d + d^2 + \cdots + d^l$, then each one will be fully contained. Furthermore, each one will be contained in exactly 1 layer. Since we have $l+1$ layers, then this will exactly be $\log(d^l) + 1$. Furthermore if a number n is between $d^l < n < d^{l+1}$, then it will half occupy the l-th layer. Therefore the most efficent, and lower bounded, packing is log. A number must be in the partial sum $\sum_{i=0}^{n} d^0$. This is exactly equal to $\frac{d^{n+1}-1}{d-l}$. Therefore for a given number n, one can find the largest l such that $\frac{d^{l+1}-1}{d-l} < n$. Then one can find the smallest tree height via $\log_d(\frac{d^{l+1}-1}{d-l})$.

1.4
- To show that $\log(n!) = O(n \log n)$, note that $n! = 1 \cdot 2 \cdots n \leq n \cdot n \cdots n$ where it's n n's, since $i \leq n$ for all $i \in [n]$. Since log is increasing then $\log(n!) \leq \log(n^n) = n \log(n)$. Therefore $\log(n!) = O(n \log n)$.
- To show that $\log(n!) = \Omega(n \log n)$, note that $(n/2)^{n/2} \leq n!$ since $(n/2)^{n/2} \leq \frac{n}{2}(\frac{n}{2}+1) \cdots n \leq n!$. Therefore $\frac{n}{2}(\log(n) - \log(2)) \leq \log(n!)$. Since this is a constant factor off of $n \log n$, and $\frac{\log 2}{n \log n} \to 0$ as $n \to \infty$ implies that $\log(n!) = \Omega(n \log n)$.

Since $\log n! = O(n \log n) = \Omega(n \log n)$ implies that $\log n! = \Theta(n \log n)$.

1.10 Suppose $a \equiv b \mod N$, then there exists $k \in \mathbb{Z}$ such that $kN = b - a$. Since $M \mid N$, then there exists $n \in \mathbb{Z}$ such that $nM = N$, therefore $(kn)M = b - a$. Thus $M \mid b - a$ by definition.

1.16 If $b = 2^l$, then $a^b = a^{2^l} = (a^2)^l \mod c$. Therefore the number of multiplications needed is $\log(b) + 1$.

1.18
- Factorization: $210 = 2 \cdot 3 \cdot 5 \cdot 7$, $588 = 2^2 \cdot 3 \cdot 7^2$. Note that $\gcd(a, b) = p_1^{\min(\alpha_1, \beta_1)} \cdots p_n^{\min(\alpha_n, \beta_n)}$ where $a = p_1^{\alpha_1} \cdots p_n^{\alpha_n}, b = p_1^{\beta_1} \cdots p_n^{\beta_n}$. Therefore $\gcd(a, b) = 2 \cdot 3 \cdot 7 = 42$

- Euclidean algorithm:

$$588 = 2 \cdot 210 + 168$$
$$210 = 1 \cdot 168 + 42$$
$$168 = 4 \cdot 42 + 0$$

Therefore 42 is the gcd.

1.18e Extended Euclidean algorithm:

$$\begin{aligned} 42 &= (210 - 168) \\ &= (210 - (588 - 2 * 210)) \\ &= 3 * 210 + (-1) * 588 \end{aligned}$$

Note that this isn't exactly the algorithm, but the algebra which the algorithm abstracts away and recurses through.

1.26 We want to compute $17^{17^{17}} \mod 10$. Since $\phi(10) = \phi(2)\phi(5) = 4$ then we need to figure out what $17^{17} \mod 4$ is. Note that $\phi(4) = 4 - 2$ therefore we have to compute $17 \mod 2 \equiv 1$. Thus $17^{17} \mod 4 \equiv 17 \mod 4 \equiv 3 \mod 4$. Therefore $17^{17^{17}} \equiv 17^3 \mod 10$. Note that $17^3 \equiv (17 \mod 10)^3 \equiv 7^3 = 343 \equiv 3 \mod 10$. Therefore $17^{17^{17}} \equiv 3 \mod 10$.