1. Professor George O’Jungle has a 27-node binary tree, in which every node is labeled with a unique letter of the Roman alphabet or the character &. Preorder and postorder traversals of the tree visit the nodes in the following order:


(a) List the nodes in George’s tree in the order visited by an inorder traversal.
(b) Draw George’s tree.
2. (a) [5 pts] Solve the following recurrences. State tight asymptotic bounds for each function in the form $\Theta(f(n))$ for some recognizable function $f(n)$. Assume reasonable but nontrivial base cases. If your solution requires a particular base case, say so. Do not submit proofs—just a list of five functions—but you should do them anyway, just for practice.

- $A(n) = 10A(n/5) + n$
- $B(n) = 2B\left(\left\lceil \frac{n+3}{4} \right\rceil \right) + 5n^{6/7} - 8\sqrt{\frac{n}{\log n}} + 9\left\lfloor \log^{10} n \right\rfloor - 11$
- $C(n) = 3C(n/2) + C(n/3) + 5C(n/6) + n^2$
- $D(n) = \max_{0<k<n} (D(k) + D(n-k) + n)$
- $E(n) = \frac{E(n-1)E(n-3)}{E(n-2)}$ [Hint: Write out the first 20 terms.]

(b) [5 pts] Sort the following functions from asymptotically smallest to asymptotically largest, indicating ties if there are any. Do not submit proofs—just a sorted list of 16 functions—but you should do them anyway, just for practice.

Write $f(n) \ll g(n)$ to indicate that $f(n) = o(g(n))$, and write $f(n) \equiv g(n)$ to mean $f(n) = \Theta(g(n))$. We use the notation $\lg n = \log_2 n$.

\[
\begin{array}{cccc}
n & \lg n & \sqrt{n} & 3^n \\
\sqrt{\lg n} & \lg \sqrt{n} & 3^{\sqrt{n}} & \sqrt{3^n} \\
3^{\lg n} & \lg(3^n) & 3^{\lg \sqrt{n}} & 3^{\sqrt{\lg n}} \\
\sqrt{3^{3\lg n}} & \lg(3^{\sqrt{n}}) & \lg 3^n & \sqrt{\lg(3^n)}
\end{array}
\]

3. Suppose you are given a pointer to the head of singly linked list. Normally, each node in the list has a pointer to the next element, and the last node's pointer is NULL. Unfortunately, your list might have been corrupted (by a bug in somebody else's code, of course), so that some node's pointer leads back to an earlier node in the list.

Describe an algorithm\footnote{Since you understand the course policies, you know what this phrase means. Right?} that determines whether the linked list is corrupted or not. Your algorithm must not modify the list. For full credit, your algorithm should run in $O(n)$ time, where $n$ is the number of nodes in the list, and use $O(1)$ extra space (not counting the list itself).
4. (a) Prove that any integer (positive, negative, or zero) can be written in the form $\sum_i \pm 3^i$, where the exponents $i$ are distinct non-negative integers. For example:

\[
42 = 3^4 - 3^3 - 3^2 - 3^1 \\
25 = 3^3 - 3^1 + 3^0 \\
17 = 3^3 - 3^2 - 3^0
\]

(b) Prove that any integer (positive, negative, or zero) can be written in the form $\sum_i (-2)^i$, where the exponents $i$ are distinct non-negative integers. For example:

\[
42 = (-2)^6 + (-2)^5 + (-2)^4 + (-2)^0 \\
25 = (-2)^6 + (-2)^5 + (-2)^3 + (-2)^0 \\
17 = (-2)^4 + (-2)^0
\]

[Hint: Don't use weak induction. In fact, never use weak induction.]

5. An arithmetic expression tree is a binary tree where every leaf is labeled with a variable, every internal node is labeled with an arithmetic operation, and every internal node has exactly two children. For this problem, assume that the only allowed operations are $+$ and $\times$. Different leaves may or may not represent different variables.

Every arithmetic expression tree represents a function, transforming input values for the leaf variables into an output value for the root, by following two simple rules: (1) The value of any $+$-node is the sum of the values of its children. (2) The value of any $\times$-node is the product of the values of its children.

Two arithmetic expression trees are equivalent if they represent the same function; that is, the same input values for the leaf variables always leads to the same output value at both roots. An arithmetic expression tree is in normal form if the parent of every $+$-node (if any) is another $+$-node.

![Three equivalent expression trees. Only the third is in normal form.]

Prove that for any arithmetic expression tree, there is an equivalent arithmetic expression tree in normal form.
*6. [Extra credit] You may be familiar with the story behind the famous Tower of Hanoï puzzle:

At the great temple of Benares, there is a brass plate on which three vertical diamond shafts are fixed. On the shafts are mounted n golden disks of decreasing size. At the time of creation, the god Brahma placed all of the disks on one pin, in order of size with the largest at the bottom. The Hindu priests unceasingly transfer the disks from peg to peg, one at a time, never placing a larger disk on a smaller one. When all of the disks have been transferred to the last pin, the universe will end.

Recently the temple at Benares was relocated to southern California, where the monks are considerably more laid back about their job. At the “Towers of Hollywood”, the golden disks have been replaced with painted plywood, and the diamond shafts have been replaced with Plexiglas. More importantly, the restriction on the order of the disks has been relaxed. While the disks are being moved, the bottom disk on any pin must be the largest disk on that pin, but disks further up in the stack can be in any order. However, after all the disks have been moved, they must be in sorted order again.

Describe an algorithm that moves a stack of n disks from one pin to the another using the smallest possible number of moves. Exactly how many moves does your algorithm perform? [Hint: The Hollywood monks can bring about the end of the universe considerably faster than their Benaresian counterparts.]