1. List and explain the steps used to analyze the time complexity of a recursive algorithm (provide as much detail as possible).

Ans:

a) Write a recurrence relation for $T(n)$, the worst-case number of steps taken by the algorithm on inputs of size n. This involves defining n as an expression of the algorithm's parameters, then analyzing the number of steps taken by the algorithm for various input sizes– with cases based on the algorithm's base cases and general cases.

b) Find upper and lower bounds on the values of $T(n)$, i.e., functions $L(n)$ and $U(n)$ such that $T(n) \in \Omega(L(n))$ and $T(n) \in O(U(n))$. Ideally, $L(n) = U(n)$; otherwise, make them as close as possible to each other.

   In the absence of other information, the "repeated substitution" method can be used to find a reasonable "guess". Start with an expression for $T(n)$ from the recurrence relation, with simplifying assumptions (like ignoring floors and ceilings), and repeatedly expand (substitute) into expressions $T(\cdots)$ on the right-hand side until the pattern becomes obvious. From the expression after k substitutions, work out how many substitutions are necessary to reach a base case, plug in that value for k, and simplify.

c) Prove, usually by complete induction, $T(n) \in \Omega(L(n))$ and $T(n) \in O(U(n))$ for the bounds $L(n)$ and $U(n)$ found in the previous step. If both bounds are simply obtained from the result of performing repeated substitution, you may need to make small modifications to the exact expression used (e.g., changing constant factors or adding low-order terms) so that you can do the proof.

Note that the final answer, the one that really matters, is the proof of each bound on $T(n)$ (step c above). The expression obtained by repeated substitution is only an approximation and should be considered as a "starting point" for these proofs. $\square$

2. Consider the following recurrence relation.

$$T(n) = \begin{cases} 2 & \text{if } n = 0, 1 \\ T(\lfloor \frac{2n}{3} \rfloor) + \log n & \text{if } n > 1 \end{cases}$$

Find upper and lower bounds on the value of $T(n)$, using the Master Theorem. Then try to find a tight bound.

Ans: This is a trick question, because the Master Theorem only applies to recurrences where the extra work performed outside the recursive calls takes time $\Theta(n^d)$ for some constant d. Unfortunately, $\log n$ is NOT such a function.

Nevertheless, we can use the Master Theorem to obtain separate upper and lower bounds on the value of $T(n)$. Consider the following recurrence relations.

$$L(n) = \begin{cases} 2 & \text{if } n = 0, 1 \\ L(\lfloor \frac{2n}{3} \rfloor) + 1 & \text{if } n > 1 \end{cases}$$

$$U(n) = \begin{cases} 2 & \text{if } n = 0, 1 \\ U(\lfloor \frac{2n}{3} \rfloor) + n & \text{if } n > 1 \end{cases}$$

Since $1 \leq \log n \leq n$ for all $n > 1$, it is easy to see that for all n, $L(n) \leq T(n) \leq U(n)$.

Applying the Master Theorem to $L(n)$, we get $a = 1$, $b = \frac{3}{2}$ and $d = 0$ so $a = b^d$ and $L(n) \in \Theta(n^d \log n) = \Theta(\log n)$. Hence, $T(n) \in \Omega(\log n)$. In this case, it would be trivial to prove $T(n) \geq \log n$ by induction on $n \geq 1$.

Remark 1. *In you proof, you may think that the induction hypothesis is not necessary, but it is to ensure that* $T(\lfloor \frac{2n}{3} \rfloor)$ *is not negative.*

Applying the Master Theorem to $U(n)$, we get $a = 1$, $b = \frac{3}{2}$ and $d = 1$ so $a < b^d$ and $U(n) \in \Theta(n^d) = \Theta(n)$. Hence, $T(n) \in O(n)$. Again, it is trivial to prove $T(n) \leq 2n$ by induction on $n \geq 1$ (use the fact that $\log n \leq n$ for all $n \geq 1$).

Unfortunately, we have reached the limit of what can be learned by applying the Master Theorem– there is no way to make the bounds any tighter.

Remark 2. *If we remember that* $\log n$ *is* $o(n^c)$ *for any realy number* $c > 0$ *(from calculus), then we can prove that* $T(n)$ *is* $O(n^c)$ *for any real number* $c > 0$*. But that still leaves a gap between the upper and lower bounds.*

In order to get a tight bound, we need to use something like the repeated substitution method. We will need to use the following facts about logarithms.
Repeated substitution for $T(n)$:

$$\begin{aligned}
T(n) &= T(\frac{2n}{3}) + \log n \\
&= T(\frac{2(\frac{2n}{3})}{3}) + \log\frac{2n}{3} + \log n \\
&= T((\frac{2}{3})^2 n) + 2\log n + \log\frac{2}{3} \\
&= T((\frac{2}{3})^3 n) + \log(\frac{2}{3})^2 n + 2\log n + \log\frac{2}{3} \\
&= T((\frac{2}{3})^3 n) + 3\log n + 3\log\frac{2}{3} \\
&= T((\frac{2}{3})^4 n) + \log((\frac{2}{3})^3 n) + 3\log n + 3\log\frac{2}{3} \\
&= T((\frac{2}{3})^4 n) + 4\log n + 6\log\frac{2}{3}
\end{aligned}$$

Educated Guess: After i substitutions,

$$T(n) = T((\frac{2}{3})^i n) + i\log n + (i\frac{(i-1)}{2})\log\frac{2}{3}$$

$T(\cdot)$ disappears from the right-hand side when $(\frac{2}{3})^i n = 1$, i.e., $i = \log_{\frac{2}{3}}\frac{1}{n} = \log_{\frac{3}{2}} n$. For this value of i,

$$T(n) = T(1) + \log_{\frac{3}{2}} n \log n + (\log_{\frac{3}{2}} n(\log_{\frac{3}{2}} n - 1)/2)\log\frac{2}{3}$$

Based on this, we expect that $T(n) \in \Theta(log^2 n)$. Proving this, however, becomes a little tricky. $\square$