1. Consider the following generic “divide-and-conquer” code:

```java
foo(A[1..n]) {
    if (n == 1) {
        return 1
    } else {
        if (...) {
            x = foo(A[1..n/3])
        } else {
            y = foo(A[1..n/3])
            z = foo(A[2n/3..n])
        }
    }
    return x + y * z
}
```

Develop a recurrence $T(n)$ for the worst-case runtime of this code, $T(n)$. Then, use the Master theorem to analyse the asymptotic runtime of $\text{foo}$.

You should have used Case 1 of the Master theorem. For each of the following, modify the above code to satisfy the condition:

- The new code should use Case 2 of the Master theorem. Perform the analysis.
- The new code should use Case 3 of the Master theorem. Perform the analysis.
- The new code cannot be analysed with the Master theorem. Explain why not.

Note: there are many possible ways to do the above. Try to see how many you can figure out!

2. For each of the following recurrences, explain why the Master Theorem cannot be used.

(a) $T(n) = nT(n/2) + 2$
(b) $T(n) = T(n - 1) + 3n^2$
(c) $T(n) = T(n/2) + \log n$ (Oops; this one can be solved using the Master Theorem!)
(d) $T(n) = 2T(n/4) + 3T(n/3) + 1$
(e) Finding a closed form expression for $T(n) = 2T(n/2) + 4n$