TREE RECURSION

Question 1. Consider the following list-of-lists (also known as a tree):

```
(define my-tree '(1 (2 3) (4 (5 6))))
```

which is equivalent to:

```
(define my-tree (list 1 (list 2 3) (list 4 (list 5 6))))
```

Using a box and pointer diagram, draw this data structure. Make sure to include the symbol `my-tree` in your drawing, with an arrow-line pointing from the symbol to the data structure.
Consider the following abstracted procedure for performing accumulations on trees:

\[
\text{(define (accumulate-tree tree term combiner null-value)}
\begin{align*}
&\text{ (cond ((null? tree) null-value))} \\
&\text{ ((not (pair? tree)) (term tree)))} \\
&\text{ (else (combiner (accumulate-tree (car tree) term combiner null-value))} \\
&\text{ (accumulate-tree (cdr tree) term combiner null-value))))}}
\end{align*}
\]

**Question 2.** Write an expression using `accumulate-tree` which *adds the leaves of my-tree*, producing 21.

**Question 3.** Write an expression using `accumulate-tree` which *scales each leaf of my-tree* by 3, producing a new tree, `'(3 (6 9) (12 (15 18)))`.

**Question 4.** Write an expression using `accumulate-tree` which adds *counts the leaves of my-tree*, producing 6.

**Other challenges:** count the # of odd leaves; output a new tree with leaves with even-numbered leaves replaced by 1s (other leaves preserved intact); output the maximum leaf value.