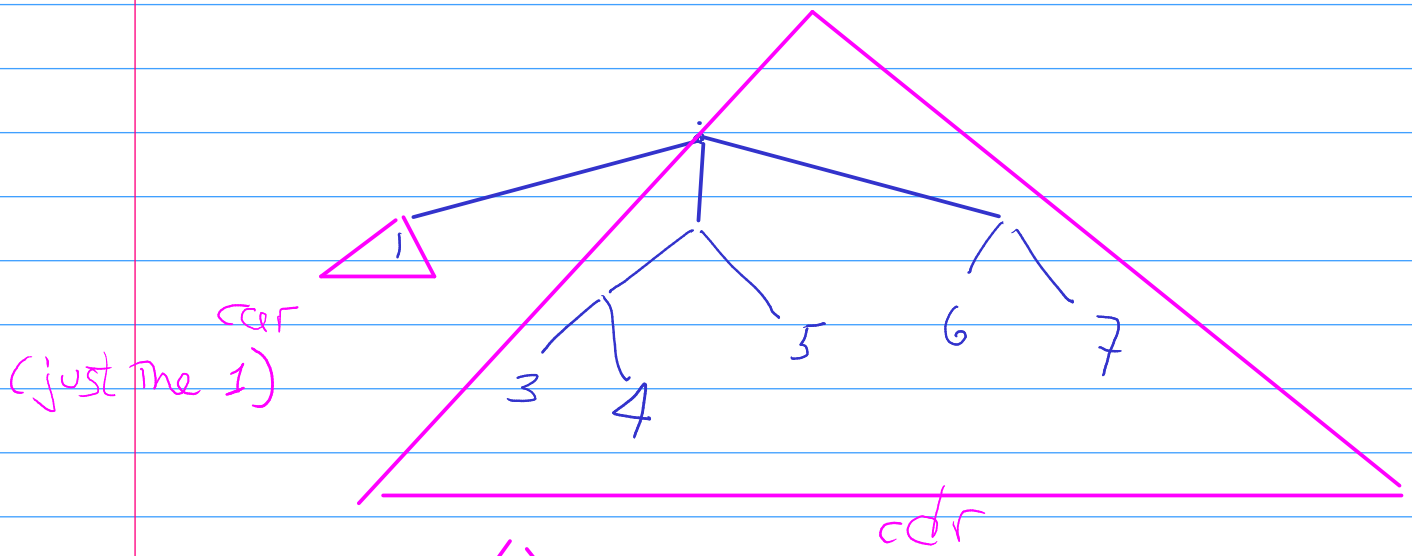
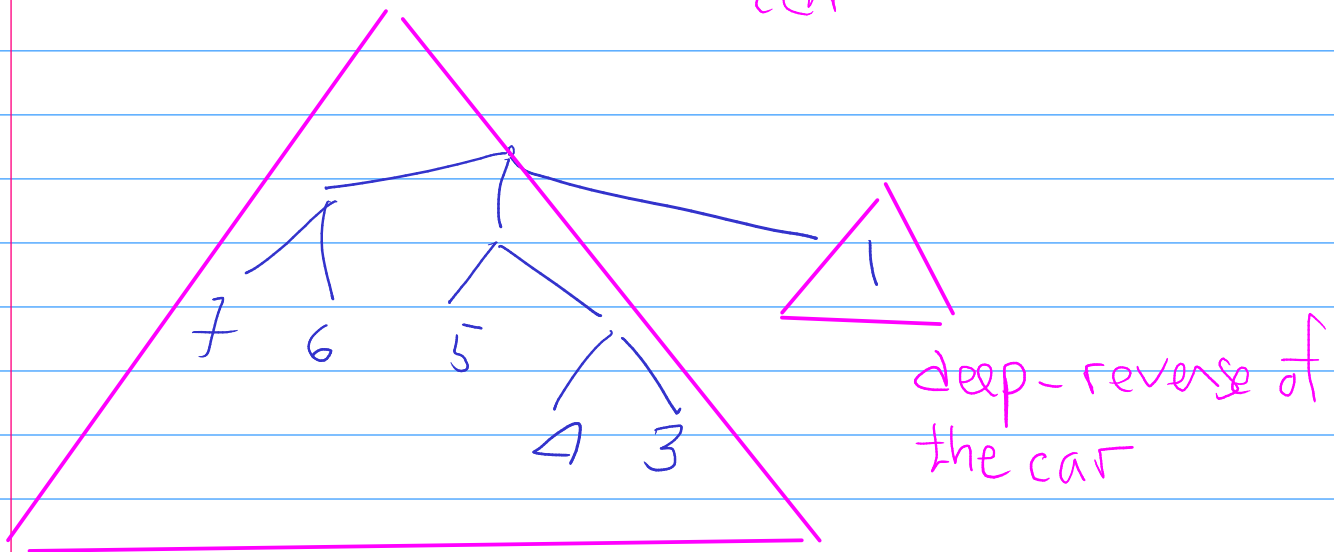
A&S Exercise 2.27, from HW 9

Deep Reverse Illustrated

```
(define (deep-reverse x)
  (cond ((null? x) '())
        ((not (pair? x))  x)
        (else (append (deep-reverse (cdr x))
                      (list (deep-reverse (car x)))))))
```





recall (eg) (append '(1 2 3) 4) =

(1 2 3 . 4)

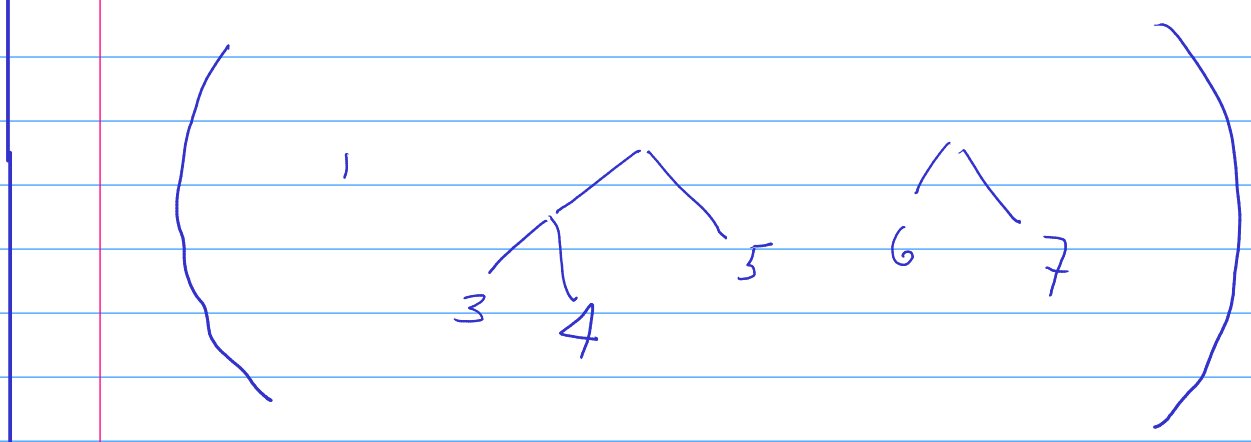
(or however r5rs presents this)

which is not a list

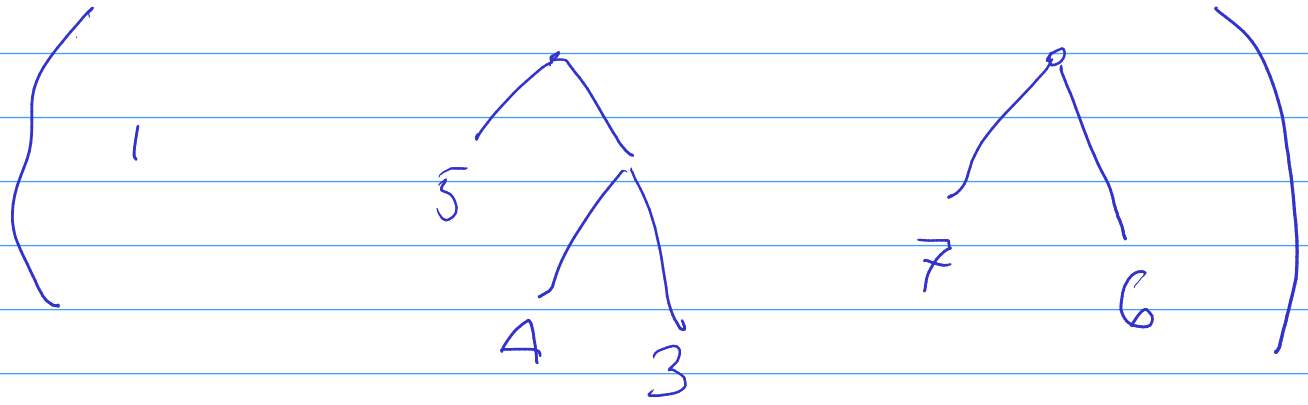
So you see that (list (car x)) is forced by append and by the logic underlying our tree diagrams — we need to insert the link terminating at 1.

We use append in the first place to preserve the root outdegree of the cdr.

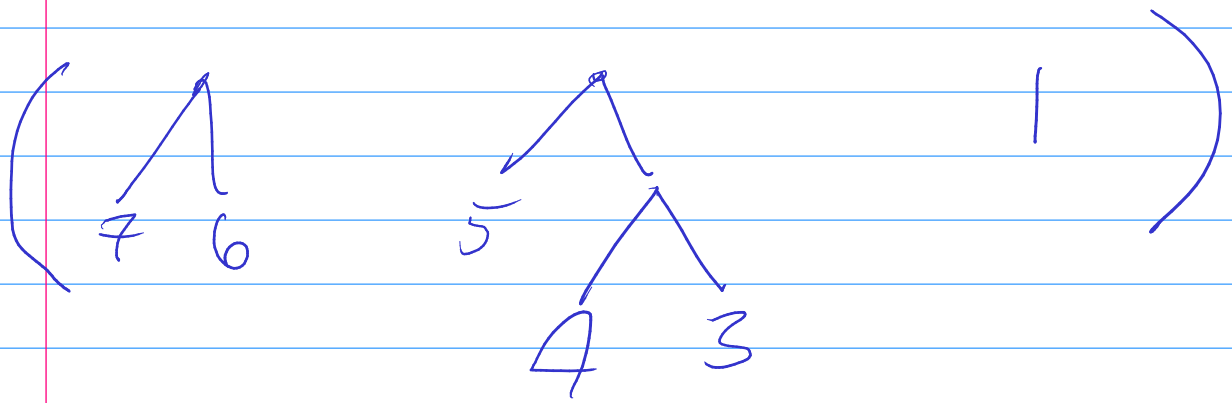
```
; the following is surely simpler to understand (and to prove
; correct) -- now looking at the argument as a list
; of subtrees

(define (alt-alt-deep-reverse tr)
    (cond ((null? tr) '())
          ((atom? tr) tr)
          (else (reverse (map alt-alt-deep-reverse tr)))))
```



first: map the deep-rev. fun'tion over t:



Giving a list of deeply reversed subtrees, but in the wrong order. How wrong? Well, its exactly the reverse of what we want. And that accounts for the call to reverse; giving



which is the list-of-subtrees view of the desired result.

We also discussed A&S exercises 2.31 (which needed 2.30) and 2.32. Again, these are part of HW 9.

I ask you to work these out on your own, making note of where you get stuck, if in fact you do get stuck. Please bring your questions either to my office hours after the break, or to one of these zoom sessions (also after the break).

I can go back to HW 7 if anyone has questions.

Of course we need to discuss further problems in HW 8, the rest of HW 9, and the later HW assignments.

This material will be emphasized heavily on both Quiz 3 and the final exam - please do not let it slide.