What Will Scheme Print?

```
1. (list (cons 3 4) (append '(the cow) '(ran)) '())
2. (cons (list (cons '(a b) 'c) '(d e)) '(f g))
3. ((lambda (+ /) (let ((- *) (/ +)) (- / +))) 2 5)
4. (caddr (cons '(3 a (cons 2 1)) '((cons 3 b) lol)))
```

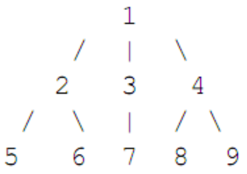
Box-and-Pointer

```
(append (list (list '(1 2))) (cons '(3) '(4 5)))
```

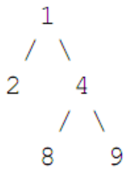
Trees

Write a procedure `burned-tree?` that takes in two Trees and returns true if the first tree is a burned-down version of the second.

Consider the following two Trees:



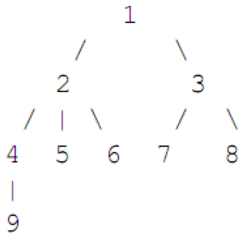
Tree A



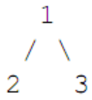
Tree B

We say that Tree B is a **burned-down version** of Tree A because both Trees have the same root and Tree B has the same structure (as in every node has same children in the same order) as Tree A except certain sub-trees got destroyed like the leaves 5, 6 and the branch 3--7.

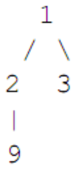
As another example:



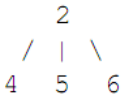
Tree C



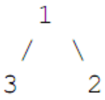
Tree D



Tree E



Tree F



Tree G

Here, Tree D is a **burned-down version** of Tree C. Tree E is NOT a **burned-down version** of Tree C because the node 2 does not have 9 has a direct child in Tree C. Tree F is NOT a **burned-down version** of Tree C because they have different roots. Tree G is NOT a **burned-down version** of Tree C because order of the direct children of the root node 1 is wrong. **Note also** that the **burned-down version** might have less children than the original Tree.

Recursion and Higher-Order Functions

Write a procedure called `product-list` that takes in a list of numbers $(a_1, a_2, a_3, \ldots)$ and returns another list $(a_1, a_1a_2, a_1a_2a_3, \ldots)$. You may use higher-order functions and recursion.

Orders of Growth

Assume a cube of water of side length *n*. Assume that we need to perform a calculation on every water molecule that takes *n* time.

1. What is the order of growth of the total number of calculations required?
2. What if we considered a sphere of water of radius *n*?