Trees
1. Write (square-tree tree), which returns the same tree structure, but with every element squared. Dont use map!

2. Write (max-of-tree tree) that does the obvious thing. The tree has at least one element.

3. Write (listify-tree tree) that turns the tree into a list in any order. (This one you cant use map even if you tried... Muwahahaha)

4. A maximum heap is a tree whose childrens data are all less-than-or-equal-to the roots datum. Of course, its children are all maximum heaps as well. Write (valid-max-heap? tree) that checks if this is true for a given tree.
Binary Search Trees
1. Jimmy the Smartass was told to write (valid-bst? bst) that checks whether a tree satisfies the binary-search-tree property: elements in left subtree are smaller than datum, and elements in right subtree are larger than datum. He came up with this:

\[
\text{(define (valid-bst? bst)}
  \text{(cond ((null? bst) \#t))}
\text{(else)}
  \text{(and (or (null? (left-branch bst))}
  \text{(and (< (datum (left-branch bst)) (datum bst))}
  \text{(valid-bst? (left-branch bst))))}
\text{(or (null? (right-branch bst))}
  \text{(and (> (datum (right-branch bst)) (datum bst))}
  \text{(valid-bst? (right-branch bst))))))}
\]

Why will Jimmy never succeed in life? Give an example that would fool his pitiful procedure.

2. Write (sum-of bst) that takes in a binary search tree, and returns the sum of all the data in the tree.

3. Write (max-of bst) that takes in a binary search tree, and returns the maximum datum in the tree. The tree has at least one element.

4. Write (listify bst) that converts elements of the given bst into a list. The list should be in NON-DECREASING ORDER!

5. Write (remove-leaves bst) that takes in a bst and returns the bst with all the leaves removed.

6. Write (height-of tree) that takes in a tree and returns the height: the length of the longest path from the root to a leaf.