## Task 2:

The rules for this program are as follows:

Rule 1 – If the recursion reaches a leaf, then stop

Rule 2 – If the right subtree is a leaf node, then recurse to the left subtree,

Rule 3 – Left rotate and re-apply LeftToRight to the current node

The LeftToRight program descends one level of the tree only when the right subtree is a leaf node.

Thus, when the program is currently working on a certain node x in the tree, the right subtree of every ancestor y of x is a single leaf node.

When Rule 3 is applied, the current tree node is not a leaf node. Also, left rotate will not turn the current node into a leaf node. So, after Rule 3 is applied, either we re-apply Rule 3, or Rule 2 is applied as Rule 1 is not applicable from the previous analysis. But we will not repeatedly apply Rule 3 forever since it is assumed that the program halts. When Rule 2 is applied, the right subtree is a single leaf node, and the recursion will continue at a lower level of the tree. The program halts when the node reached is a leaf node z. By the previous discussions, the right subtree of every ancestor y of z is a single leaf node. Therefore, the tree is a standard left-to-right evaluation tree.

## Task 3:

Recall that if a tree consists of a single leaf node, the size of the tree is 1, otherwise the tree size is size(left subtree) + 2 * size(right subtree). We want to prove the lemma that for all tree t, size(t) > 0 by a structural induction on the trees.

Base case: When the tree t is a single leaf node tree.

By definition, the size(t) = 1, which is bigger than 0.

Hypothesis: suppose size(t1) > 0 and size(t2) > 0.

Induction:

Consider a tree with left subtree t1 and right subtree t2.

Size(t) = size(t1) + 2 * size($t_2$) (by the definition of size)

> 0 + 2 * size($t_2$) (by induction hypothesis; size(t1) > 0)

= 2 * size($t_2$)

> 2 * 0 (by induction hypothesis; size($t_2$) > 0)

= 0

Therefore, the lemma that for all tree t, size(t) > 0, is proved.

In the following, we want to prove that every time the logic of LeftToRight is recursively applied, the size measure of the new tree argument in the recursive call is reduced. According to the logic of LeftToRight, the program recurses under Rules 2 and 3.

Case 1: Rule 2.

From the current tree t, the program recurses to the left subtree t1 when the right subtree t2 is a leaf node.

We want to show that size(t1) < size(t). By definition, size(t) = size($t_1$) +2 * size(t2) and by the lemma about tree sizes, we have size($t_2$) > 0. Therefore, size(t) > size($t_1$) + 2 * 0 = size(t1).

Case 2: Rule 3.

The current tree has a left subtree t1 and the right subtree consists of two subtrees t2 and t3. Then after a left rotation, the recursion is applied to a tree with left subtree consisting of subtrees t1 and t2, and right subtree t3.

Now show that size(new tree) < size(old tree). By the definition of size, we want to show that

(size(t1) + 2 * size(t2)) + 2 * size(t3) < size(t1) + 2 * (size(t2) + 2 * size(t3))

= size(t1) + 2 * size(t2) + 2 * size(t3) < size(t1) + 2 * size(t2) + 4 * size(t3)

= 2 * size(t3) < 4* _ size(t3)

= 0 < 2 * size(t3)

This is true by the lemma that size(t3) > 0.

## Task 4:

The proof is by an induction on the size of a tree. Note that the size of a tree is 1 for a leaf, and size(left subtree) + 2 _ size(right subtree) otherwise.

Base case: When the size of a tree t is 1.

The tree t is a single-node tree, which is a left-to-right evaluation tree.

Hypothesis: Suppose for all trees $t^n$ of sizes less than s, where

s * 2, (LeftToRight $t^n$) returns a left-to-right evaluation tree.

Induction: We want to show that for any tree t, if (size t) = s, then (LeftToRight t) is a left-to-right evaluation tree.

Case 1: there is no such tree t where (size t) = s.

As the premise of the if-statement that we need to prove does not hold, the if-statement is verified.

Case 2: there is a tree t where (size t) = s * 2.

As (size t) > 1, the tree is not a leaf node tree. So, Rule 1 does not apply.

Case 2.1: Rule 2 applies.

The tree returned is

Node (LeftToRight $t_1$) (Leaf n)

As (size $t_1$) < (size t) = s, by the induction hypothesis, (LeftToRight $t_1$) returns a left-to-right evaluation tree $t^n$. Extending the left-to-right tree $t^n$ with another right node (Leaf n), the result is still a left-to-right tree.

Case 2b: Rule 2 does not apply, but Rule 3 applies.

Let tree t be (Node t1 (Node t2 t3))

A left rotate operation is applied to t to turn it to $t^n$ = (Node (Node t1 t2) t3)

By the answer from Task 3, (size $t^n$) < (size t) = s. The induction hypothesis applies to (LeftToRight $t^n$) which returns a left-to-right evaluation tree.