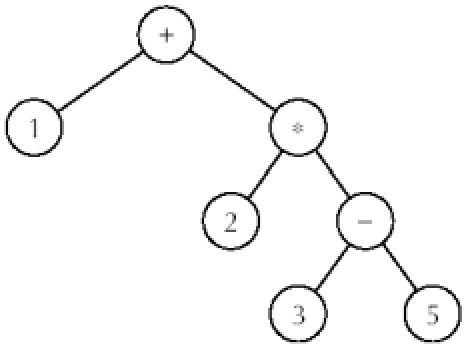
Activity 8

Consider the data type:

```
data ExprTree = Leaf Float | Add ExprTree ExprTree | Sub ExprTree ExprTree
                | Mul ExprTree ExprTree | Div ExprTree ExprTree deriving Show
```

The expression: (Add (Leaf 1) (Mul (Leaf 2) (Sub (Leaf 3) (Leaf 5))))
represents 1+2*(3-5).

We can view this expression by drawing a tree diagram:



Write expressions using the ExprTree data type to represent the following:

1 + 3 * 5 ____

(1 + 3) * 5 ____

This shows that the tree can represent the precedence of the operators, or in other words it describes which operator gets evaluated first.

Draw a tree diagram that illustrates each of the expressions above.

Now examine the function:

```
evaluate :: ExprTree -> Float
evaluate (Leaf x) = x
evaluate (Add e1 e2) = evaluate e1 + evaluate e2
evaluate (Sub e1 e2) = evaluate e1 - evaluate e2
evaluate (Mul e1 e2) = evaluate e1 * evaluate e2
evaluate (Div e1 e2) = evaluate e1 / evaluate e2
```

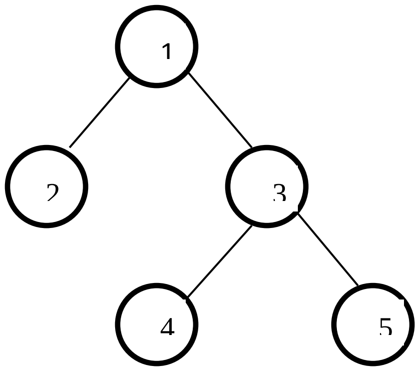
For each of the nodes in the tree, Add, Sub, Mul, and Div, we recursively call evaluate on the two "children" trees e1 and e2.

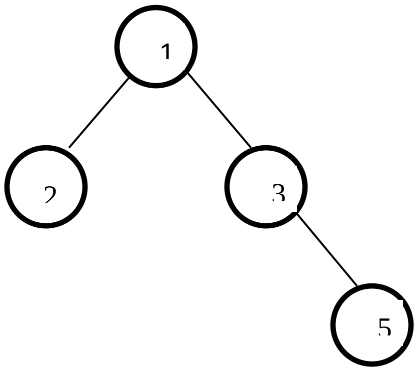
Write up the substitution models for evaluate on the two expressions that you wrote up above. Do you see the "tree recursion"? Tree recursion just means that we do the recursion for each of the children and combine the two results in some way.

Consider the data type:

data Tree a = Empty | Node a (Tree a) (Tree a) deriving Show

Write expressions using this data type for the following trees:





Let's define a *complete* tree as one in which every node either is a leaf or that node has two children. That would mean that the first tree above is complete and the second tree is not.

Write a function which returns true if a tree is complete and false otherwise:

```
complete :: Tree a -> Bool
complete Empty = True
complete (Node x Empty Empty) = ___________________
complete (Node x t1 Empty) = __________________
complete (Node x Empty t2) = __________________
complete (Node x t1 t2) = ________________________________________________
```

Test your function on the tree expressions that you wrote above.