1. A binary tree is of the form – a node, its left child node, and its right child node.
    a. • Example Node = Node(value, leftnode, rightnode);
    b. • Example Tree = Node(’a’, Node(’b’, Empty, Empty), Node(’c’, Empty, Empty)).

    This binary tree has a root node with value ‘a’, which has two children with values ‘b’ and ‘c’ that are leaf nodes. Write a function in OCaml to count the number of leaves of a binary tree.

2. Think of a binary tree to represent basic arithmetic operations. In such a tree, the leaf nodes will be of a numeric type while the internal non-leaf nodes will hold the arithmetic operations of addition, subtraction, multiplication, and division. This means, we need to define a binary, tree data type over two distinct data types. Define such a binary tree in OCaml.

3. Next, write down a function that takes a binary tree of the type you defined, and returns a tuple with its first element being a list of operators, and its second element being a list of the numeric values in the nodes. That is, the function’s type should be val function name : ('a, 'b) tree -> 'b list * 'a list = <fun> where 'a is the operator’s type and 'b is the numeric type.