Assignment: Advanced Programming

Course code: IT712A

Vicenç Torra

1 Assignment Number 2

This is the second (and last) assignment for the Advanced Programming Course. Take into account the following when preparing and submitting the exercises.

- The assignment is individual.
- Deliver the solutions by email to my urkund-address, which is

  `vicenc.torra.his @analys.urkund.se`.

  with cc to our **BOTH** usual addresses:

  `vtorra@his.se` and `elio.ventocilla@his.se`

  VERY IMPORTANT:
  Use [Advanced Programming: Assignments] in the subject of the message.
- The solution should consist on two files. One text file with the code of the solution (in Scala). One PDF file with a report with a brief discussion on the solution, on how the solution has been built, and including the references used. The document should also include at least an example of the application of each function.
- Solutions have to be delivered by Oct. 20th.
- Evaluation of the written assignment (U/G).

Exercise 1. Consider the definition of an algebraic data type (binary) tree. The nodes of the tree are operations (at least two are possible "+" and "*") and the leaves are numbers (see Figure 1).

Define the data type as polymorphic so that you can use at least two types of *numbers* (e.g., a real and a complex). Then, define the following functions.

1. A function `preorder` that given a tree makes a traversal in preorder and return a list of the elements. Recall that in a preorder, we first visit/display the root and then the two subtrees.
2. A function `inorder` that given a tree makes a traversal in inorder and return a list of the elements. You may consider adding parenthesis to avoid ambiguities. Recall that traversal in inorder means that first we visit/display the first subtree, then the root and then the other subtree.
3. A function `evaluate` that given a tree evaluates it and return its corresponding value.

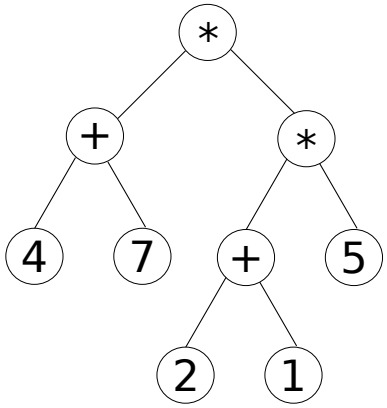

Fig. 1. Binary tree representing the expression (4+7)*((2+1)*5).

For example, a possible definition for the tree in Figure 1 as well as the application of these functions is given below. In the outcome when an integer is printed we prefix it with "i".

```
val example:Tree[Integer] = Add(mult,
  Add(add,Leaf(new Integer(4)),Leaf(new Integer(7))),
  Add(mult,Add(add,Leaf(new Integer(2)),Leaf(new Integer(1))),
           Leaf(new Integer(5))))
scala> inorder(example)
res10: List[Any] = List(i4, add, i7, mult, i2, add, i1, mult, i5)
scala> preorder(example)
res12: List[Any] = List(mult, add, i4, i7, mult, add, i2, i1, i5)
evaluate(example)
res21: Integer = i165
```

Optional exercise.

1. Consider a redefinition of the algebraic data type so that not only integers but also some variables are allowed. Define a function `evaluateWithVariables` that given a tree and a list of associations (variable → value) evaluates the tree and returns its value.