Homework Assignment #4

Due Tuesday 12/2

1. We saw an implementation of binary trees in ML (slide 74). Write a similar data structure in Cecil. That is, a tree is either the Empty tree constant or a Node consisting of an element and two subtrees. (The type of the elements will be unspecified, as in Scheme.)

More specifically, you should define an abstract Tree object with two children, a template Node object (with the appropriate fields attached and a new_node method to create concrete anonymous children of Node) and a concrete Empty object. (You do not need to write any other methods (e.g. insert, member) for your trees, although it would make good practice.)

Give a non-trivial example showing how to use your implementation to construct a tree.

2. Using your tree data structure from problem 1, implement a pre-order traversal operation, named "do", implemented by one or more overloaded "do" methods. The "do" operation takes a tree and a one-argument closure, and invokes the closure on all elements of the tree, in pre-order. The "do" operation does not return any result, unlike "map".

For example, the following should work to print out the elements of a tree:

```
let my tree := ...;
do(my_tree, &(elem){ elem.print_line; });
```

3. Write a "sum_tree" function in terms of the "do" operation in the previous question (do not check whether tree elements are numbers). Hint: you'll need to have the closure argument to "do" update a lexically-enclosing counter variable.

4. Extra credit: write a "reduce_tree" function that does not create an intermediate data structure. Use "reduce_tree" to implement "sum_tree" without using any side-effects.

5. a. Expand your tree data structure to enable sub-trees to be either binary or ternary (a ternary tree has three subtrees). Also write the code that's necessary to make "do" and "sum_tree" work for these mixed trees. You should do it by adding new code only, without modifying the existing code.

Only submit the code that needs to be added to your answers to the previous questions.

b. Is it possible to do this kind of extension in ML without modifying the existing code? How, or why not?