Homework 8

March 17, 2014

Instruction

Write the answers to all questions in one text file of the name `hwk8.rb`.

Ruby programming

This question is to define Ruby classes for representing trees of strings. The trees are made up of leaves and binary nodes. Each leaf contains a string and has no children. Each binary node has no string by itself but it contains two children, each of which can be either leave or binary node.

Define two classes `Leaf` and `BinaryNode` with the methods described below.

1. Leaf class:
   (a) `initialize` takes one argument (assumed to be a string, no need to check).
   (b) `concatAll` takes no arguments and returns the string of the leaf.
   (c) `firstAlphabetical` takes no arguments and returns the string in the leaf
   (d) `iterate` takes a closure parameter and calls the closure with the string in the leaf as argument.
2. BinaryNode class:
   (a) `initialize` takes two arguments, both of which are assumed to be either leaves or binary nodes. These are the binary node's children.
   (b) `concatAll` takes no arguments and returns a single string that is all of a tree's strings (i.e. all strings of all tree descendants) concatenated together in left-to-right order.
   (c) `firstAlphabetical` takes no arguments and returns the string in the whole tree that comes first alphabetically. You may use the `casecmp` method of the `String` class, which compares a string with the current string and returns a number great, equal, or less than zero. For example `"left".casecmp("right")` returns `-1`.

(d) `iterate` takes a closure parameter (i.e. an anonymous function) and calls the closure with each string in the tree as argument. You can create a closure using the function `lambda` and a *block* in the form of

```
lambda {|argument| block of code }
```

To invoke a closure of the name `itr` as a function, you make the call as `itr.call(some_argument)`.

Tests

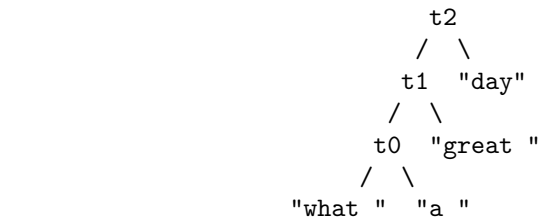
Test program:

```
def test_tree
  l0 = Leaf.new "What "
  l1 = Leaf.new "a "
  l2 = Leaf.new "great "
  l3 = Leaf.new "day"
  t0 = BinaryNode.new(l0,l1)
  t1 = BinaryNode.new(t0,l2)
  t2 = BinaryNode.new(t1,l3)

  puts "tree concatenated:\t" + t2.concatAll
  puts
  puts "tree first alphabetical:\t" + t2.firstAlphabetical
  puts
  puts "tree iterated:"
  puts
  i = 0
  t2.iterate(lambda { |s| i=i+1; puts i.to_s + ": " + s })
end

test_tree
```

In the above example, the binary node `t2` is tree of the following form.



Result of the tests:

```
E:\doc\uwm\431\2014Spring\homeworks\8>ruby hwk8.rb

tree concatenated:      What a great day

tree first alphabetical:        a

tree iterated:

1: What
2: a
3: great
4: day
```