CPSC 319: Data Structures, Algorithms and their Applications - Third assignment

---

The problem: a very customized ternary tree

The task your system has to perform is to implement a ternary tree with non-empty strings as payload and a set of operations on the tree so that the system can read in a sequence of these operations from the input file and outputs the results from those operations that produce certain outputs (in addition to often manipulating the tree).

The strings can consist of all printable characters excluding the blank (although the occurence of a particular character at the beginning of a string can change the operation that the string occurs in as argument, see below).

When the system starts, it has to create an initial tree consisting just of the root with the string
root
as payload.

In the following, a and b are variables that represent valid strings.

There are 3 operations that add nodes to the tree:

- AddL(a,b)
- AddM(a,b)
- AddR(a,b)

They add a node with payload b to the node with payload a in the tree as left child, middle child, right child, respectively. If there is no node with payload a in the tree, these operations have no effect. If there are more than one node with payload a in the tree, the operations are applied to the node with payload a that has the highest level (numerically) and if there are more than one of those nodes to the one the most right in the tree. If the correct node with payload a already has a child node at the indicated position, the operation does not change the tree, but writes
Add operation not possible.
to the output file and continues on to the next operation in the input file.

If there are more than one comma in the concatenation of a,b, then the comma intended as the comma separating the arguments a and b is the first comma, except if this comma is followed by other commas. If it is followed by other commas, then the last of this sequence of commas is the comma seperating the two arguments (and all commas before it belong to the first argument).

There are 3 operations that delete nodes from the tree:

- DelL(a)
- DelM(a)
- DelR(a)

Each operation deletes the whole subtree at the indicated position (left, middle, right) of the node with

payload a. If there is no node with payload a in the tree, these operations have no effect. If there are more than one node with payload a in the tree, the operations are applied to the node with payload a that has the highest level (numerically) and if there are more than one of those nodes to the one the most left in the tree. If the correct node with payload a has no child at the indicated position, the operation does not change the tree and the system continues on to the next operation in the input file.

There is one operation that changes the payload of nodes in the tree:

- Exchange(a,b)

This operation changes the payload of all nodes in the tree that currently have payload a to the new payload b. If there are no nodes with payload a no changes to the tree happen.

If there are more than one comma in the concatenation of a,b, then the comma intended as the comma separating the arguments a and b is the first comma, except if this comma is followed by other commas. If it is followed by other commas, then the last of this sequence of commas is the comma seperating the two arguments (and all commas before it belong to the first argument).

Finally, there is the operation

- Print()

which prints the current tree in the following manner:
the payloads of the nodes in the tree should be outputted level by level, starting with the root level, one level per line in the output. The sequence the payloads of the nodes in a level should be outputted is from left to right (and, obviously, left child, then middle child and then right child), printing each string seperated by a blank, followed by a semicolon followed by a blank. The last payload element of a level should not be followed by anything. If a node does not have nodes in the positions of all of its children, then these "missing" children do not produce any output (and there should be no seperation strings for non-existing nodes) when their level is outputted (and this recursively applies to all deeper levels).

The main customization aspect of the ternary tree is around using one specific character, namely
$
to modify some of the operations above. If the string b starts with a $, all Add operations overwrite the payload of the appropriate child node of the node identified via the a argument by the string following the $, if such a node exists. Otherwise, the operations should proceed as described above (using as b the initial b without the $; an additional $ does not create any changes and is also not eliminated).

The Exchange operation does not replace the string a by b for all applicable nodes, instead it appends the string after the $ given by b to the already existing payload a.

Input file

In an input file, there is a sequence of the commands defined above, each command in its own line. Your system has to perform each command as described, including generating the described output.

Output file

In the output file, you write the outputs described above for each command in the input file in the order as given by the input file (each starting in a new line, if output is required). If during reading of the input file, your system encounters an input that is not a valid command as defined above (or below), your system has to

output:
Input error.
and terminate.

General requirements

Regarding blanks and empty lines, the same requirements as in assignment 1 and 2 have to be fulfilled (if not stated otherwise above).