Class: S7 ICT

Date: Tuesday, April the 24th, 2023

Teacher: Mr Barsamian

B Test — With computer

Family name: _____

First name: _____

Grade: ___ / 10

Duration: 1 hour (60 minutes).

This test has to be done both on paper and on computer. At the end of the test, make sure to upload the python file on your teacher's USB key.

Some questions are bonuses, and it is highly advised to do them only at the end, when everything else has been done.

If needed, the candidate can also handle some comments inside the code or on paper.

Please keep track of the clock, and avoid spending too much time on a question. Stay focused, and good luck!

Short description of this work:

We will handle set of strings using a Tree data structure. A tree contains multiple strings. The start of each of those strings is the root of the tree, and a string ends at a node marked as an end node.

1 Introduction — 0 points

Please start by downloading the following file, that you'll have to update for this test:

http://www.barsamian.am/2022-2023/S7ICTB/BTest_Word_sets.py

A string consists of multiple characters. We will model set of strings seen in this test by trees, see Listing 1. If a tree models a set of strings, then each of its nodes will either be empty, either will contain a single character `data`, a boolean `is_end_node`, and have two children `left` and `right`. Each string of the set starts at the root of the tree, and ends at a node marked as an end node.

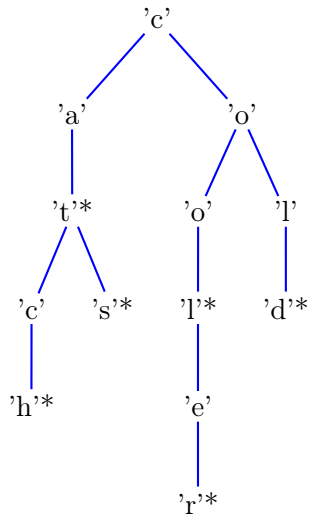
```
1  class Tree:
2      def __init__(self, data, is_end_node, left=None, right=None):
3          self.data = data
4          self.is_end_node = is_end_node
5          self.left = left
6          self.right = right
7
8      def __str__(self):
9          return str(self.data)
```

Listing 1: Python code for the Tree data structure.

The file contains Listing 2, which implements the tree on the right (end nodes are marked with a star). This tree thus holds the set of strings {cat, catch, cats, cool, cooler, cold}.

```
1  str1_node13 = Tree('r', True)
2  str1_node11 = Tree('h', True)
3  str1_node12 = Tree('e', False, str1_node13)
4  str1_node7 = Tree('c', False, str1_node11)
5  str1_node8 = Tree('s', True)
6  str1_node9 = Tree('l', True, str1_node12)
7  str1_node10 = Tree('d', True)
8  str1_node4 = Tree('t', True, str1_node7, str1_node8)
9  str1_node5 = Tree('o', False, str1_node9)
10 str1_node6 = Tree('l', True, str1_node10)
11 str1_node2 = Tree('a', False, str1_node4)
12 str1_node3 = Tree('o', False, str1_node5, str1_node6)
13 str1_node1 = Tree('c', False, str1_node2, str1_node3)
```

Listing 2: Python code for the first set of strings.



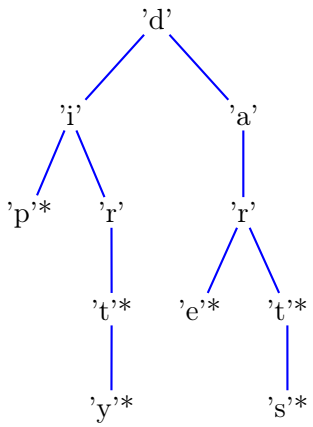
2 First steps — 4.5 points

1. The file also contains Listing 3. Please draw, on paper, a tree which would correspond to this implementation (mark the end nodes with a star).

```
str2_node13 = Tree('n', True)
str2_node12 = Tree('o', False, str2_node13)
str2_node11 = Tree('i', False, str2_node12)
str2_node8 = Tree('s', False, str2_node11)
str2_node9 = Tree('k', True)
str2_node10 = Tree('d', True)
str2_node4 = Tree('n', True, str2_node8)
str2_node5 = Tree('d', True)
str2_node6 = Tree('c', False, str2_node9)
str2_node7 = Tree('l', False, str2_node10)
str2_node2 = Tree('a', False, str2_node4, str2_node5)
str2_node3 = Tree('o', False, str2_node6, str2_node7)
str2_node1 = Tree('m', False, str2_node2, str2_node3)
```

Listing 3: Python code for the second set of strings.

2. Let us consider the tree drawn below (end nodes are marked with a star). Please write, in the Python file, a Tree which would model this set of strings.



3. In the Python file, the function `mystery` is provided (also given in Listing 4) which take as argument a tree. Please explain step by step what happens if you execute `mystery(str1_node1)`.

```
def mystery(tree):
    if tree == None:
        return ""
    elif tree.right == None:
        return tree.data + mystery(tree.left)
    elif tree.left.data < tree.right.data:
        return tree.data + mystery(tree.right)
    else:
        return tree.data + mystery(tree.left)
```

Listing 4: Python code for the mystery function.

3 Counting — 5.5 points

1. Please give a function `count_nodes` that takes as argument a tree and that counts the number of nodes in that tree. We only count the nodes that are drawn on the representations on the previous pages, we don't count `None`.

   Unit tests:

   - `count_nodes(str1_node1)` should return 13;
   - `count_nodes(str1_node2)` should return 5.

2. Please give a function `count_end_nodes` that takes as argument a tree and that counts the number of end nodes in that tree.

   Unit tests:

   - `count_spaces(str1_node1)` should return 6;
   - `count_spaces(str1_node2)` should return 3.

3. The function `mystery` that we have seen earlier takes as argument a tree, and returns the string (from the ones in the set represented by the tree) which is the biggest, with respect to the lexicographical order (with respect to the order of the dictionary). We now want to slightly modify this function to return the string which is the smallest (with respect to the lexicographical order).

   (a) What is the smallest string in the set represented in Listing 2?

   (b) What is the smallest string in the set represented in Listing 3?

   (c) Copy-paste the `mystery` function, then rename it to `smallest` and modify it in order to return the smallest string in the set.

4. A tree is correctly representing a set of strings if all its nodes have a single character in the `data` field. All trees given previously are correct, but for instance the tree defined by `Tree('catch', True)` is not correct.

   Please write a function `is_correct` that takes as argument a tree and that returns True if this tree is correct, and that returns False otherwise.

   BONUS — The data field is not necessarily a string. Verify that your function works is, for instance, the data field contains a number.