Algorithmics
Final Exam #2 (P2)

Undergraduate 1^{st} year S2#
EPITA

January, 7th 2020 - 13h-15h

Instructions (read it) :

- ☐ You must answer on **the answer sheets provided.**
  - No other sheet will be picked up. Keep your rough drafts.
  - Answer within the provided space. **Answers outside will not be marked**: Use your drafts!
  - Do not separate the sheets unless they can be re-stapled before handing in.
  - Penciled answers will not be marked.
- ☐ The presentation is negatively marked, which means that you are marked out of 20 points and the presentation points (maximum of 2) are taken off this grade.
- ☐ **Code:**
  - All code must be written in the language `Python` (no C, CAML, ALGO or anything else).
  - **Any `Python` code not indented will not be marked.**
  - All that you need (types, routines) is indicated in the **appendix** (last page)!
  - Your functions must follow the given examples of application.
- ☐ Duration : 2h

Exercise 1 (Leonardo Trees – *3 points*)

In this exercise we will study some properties of a certain type of tree: the Fibonacci trees. These are defined recursively as follows:

$$\begin{cases} A_0 = \ EmptyTree \\ A_1 = < o, EmptyTree, EmptyTree > \\ A_n = < o, A_{n-1}, A_{n-2} > \ if \ n \geqslant 2 \end{cases}$$

1. Give a graphical representation of the Fibonacci tree A_5.
2. (a) Give, in terms of $n \geqslant 2$ the height h_n of the tree A_n.
   (b) Prove that the tree A_n is height-balanced.

Exercise 2 (Leonardo Trees, again – *4 points*)

We take the Fibonacci trees defined the exercise 1, adding labels to nodes:

$$\begin{cases} A_0 = \ EmptyTree \\ A_1 = < 1, EmptyTree, EmptyTree > \\ A_n = < fibo(n), A_{n-1}, A_{n-2} > \ if \ n \geqslant 2 \end{cases}$$

Write the function `leonardo_tree(`n`)` that builds the Fibonacci tree A_n. As indicated above, the nodes will be labeled with the values of $fibo(n)$ according to the following definition:

$$\begin{cases} n \leq 1 \Rightarrow fibo(n) = \ n \\ n > 1 \Rightarrow fibo(n) = \ fibo(n-1) + fibo(n-2) \end{cases}$$

Exercise 3 (Deletion – *7 points*)

Reminder of the principle of deleting an element in a binary search tree (BST):
The structure is the same as the research.
Once the element found (it is in root), if the node that contains the element is:

- a leaf, we delete it directly: the result is the empty tree;
- a single node, we delete the node : it is replaced by its only child;
- a double node, we replace the key in root by the maximum of its left subtree, and we call again the deletion of this value on the left.

1. Write the function `maxBST(`B`)` that returns the maximum values in the non-empty BST B.
2. Write the recursive function `delBST(`B`,` x`)` that deletes the value x (the first found) in the BST B and returns the result tree.

Exercise 4 (Construction – *4 points*)

Starting with an empty tree build the AVL corresponding to the successive insertions of values 25, 60, 35, 10, 20, 5, 70, 65.
You have to draw the tree at two steps:

- after insertion of 20 ;
- the final tree.

Give used rotations in order (for instance if a left rotation occurred on the tree the root of which is 42, write *lr(42).*)

Exercise 5 (What is this? – *3 points*)

Consider the following functions:

```
        def __test(B):
            if B == None:
                return (-1, True)
            else:
                (hl, tl) = __test(B.left)
                (hr, tr) = __test(B.right)
                return (1 + max(hl, hr), tl and tr and abs(hl-hr) < 2)

        def test(B):
            (x, res) = __test(B)
            return res
```

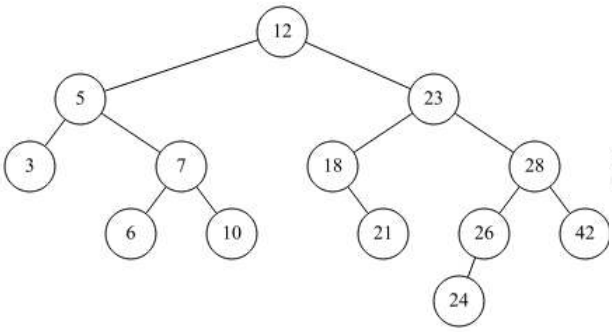

Figure 1: Tree B_1

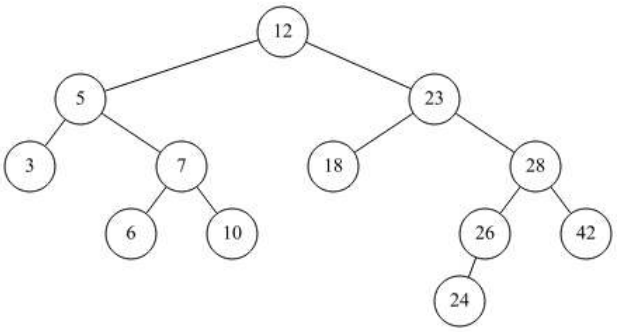

Figure 2: Tree B_2

1. For each of the above tree, what is the result returned by `test(`B_i`)` ?
2. What does the function `test(`B`)` do?
3. This function can be optimized. How?

Appendix

Binary Trees

```
class BinTree:
    def __init__(self, key, left, right):
        self.key = key
        self.left = left
        self.right = right
```

Your functions

You can write your own functions as long as they are documented (we have to known what they do).
In any case, the last written function should be the one which answers the question.