Assignment 1

Assignment 1: Lists, Queues, Trees

Implement an AVL tree (in MyAVL4StringsImpl.java), that contains String elements and that supports the following operations:

– The **constructor** method of the MyAVL4StringsImpl must not receive any parameter. That is, "public MyAVL4StringsImpl(){}" if you specify it.

– **public void insert(String element)**. //Insertion operation in AVL trees seen during the lectures. You can get full inspiration from the code in the reference book. This operation must execute in worst-case **O(log N**).

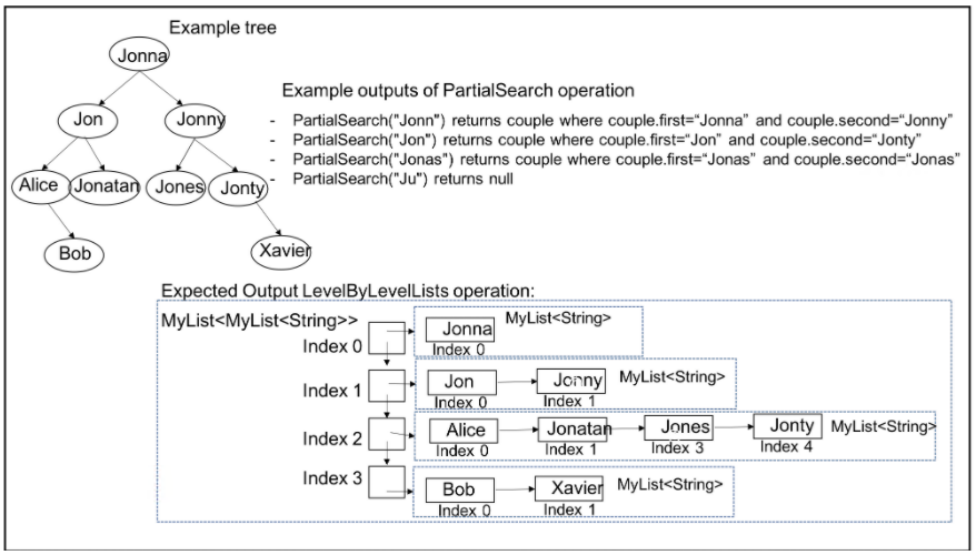
– **public Couple<String> partialSearch(String beginning) .** This operation returns the smallest and the largest elements in the tree that start with 'beginning' parameter. It returns these values in a "Couple" element (code provided in the .zip)

In the example in the figure below,

- - the partialSearch("Jonn") will return a couple where couple.first=Jonna and couple.last=Jonny;
- - The partialSearch("Jon") will return a couple where couple.first="Jon" and couple.last="Jonty".
- - The partialSearch("Jonas") will return a couple where couple.first="Jonas" and couple.last="Jonas".
- - The partialSearch("Ju") will return null.

We have provided the interface Couple and a class CoupleImpl.java that implements the interface. You can use the implementation (and you must use the interface).

This operation must execute in worst-case **O(log N)**



Important 1: You must implement your data structures from scratch without reusing data structures from libraries (none among java lists, stacks, queues, trees, etc.). This means that, at least, you will need to implement a simplified list that implements the interface Mylist<E> (code of the interface provided in the .zip).

Important 2: At least, the minimal tests provided in the .zip must pass without any modification in their code. Therefore, you must be careful with your constructor, package names, implementing the interfaces as they are provided in the .zip, etc. If these tests do not pass, we cannot provide further feedback about other problems. However, these tests are not the comprehensive set of tests for the assignment!! You should test your code yourself. You can see the tests provided in the .zip as a kick the tires (https://en.wiktionary.org/wiki/kick_the_tires) of your submission or simple smoke tests (https://en.wikipedia.org/wiki/Smoke_testing_(software)).

– **public void MyList<MyList<String>> LevelByLevelLists().** This operation returns the level by level traversal of the tree. The format of the level by level traversal is the following. The method returns a list (1) of lists (2) of Strings. Each position "d" in the list (1) contains a list (2) which has as many elements as the number of elements in the tree at depth "d". All the elements in the list (2) have to be ordered as the appear from left to right in the tree. The size of the list (1) is the number of levels in the tree (the Height of the root + 1).

This method must execute in **worst-case O(N) for full grade** in the exercise. This method can execute in worst-case O(N^2) and you can still pass the exercise.

See the example in the figure below to get a more clear view of the tasks.