UNIVERSITY OF JOHANNESBURG

Computer Science 3A

Practical Assignment 9

9 May 2024

Time: 09 May 2024 13:45 – 17:00 Marks: 50

Practical assignments must be uploaded to `eve.uj.ac.za` **before** 17h00 in the practical session.

Late submissions **will not be accepted**, and will therefore not be marked. You are **not allowed to collaborate** with any other student. You **must** upload your assignment to Eve **before** it will be marked.

AVL Trees are a type of binary search tree that allows items to be found in $O(\log n)$ time. These are self-balancing trees meaning that they will rearrange themselves to maintain the height-balance property.

You are required to complete the provided implementation for an AVL Tree and the associated application example.

You must complete the following methods marked by:

```
//COMPLETE CODE HERE
```

Please note that you should not add any additional methods to any classes provided.

The provided application example must provide an index of a text document that is searchable. This index must show the line and word number for an item in the text file. The output is expressed as `linenumber(wordnumber)` in the example below. For each word in the provided text file, you should convert the word to lowercase and ignore everything but lowercase letters. Your program should make use of the AVL Tree to output the following:

```
There are 1230 unique items in the index.
Looking up ’computer’
[1(1), 3(1), 3(22), 4(36), 4(69), 7(1), 13(4), ...
Looking up ’bob’
null
Looking up ’wikipedia’
[2(2)]
Looking up ’turing’
[77(46), 109(137)]
```

In order to solve this problem you need to complete the following functions:

1. `Main.main` — The main test programme where the indexing is done.
2. `AVLTree.restructure` — Restructure the AVL Tree according to the AVL tree restructure rules
3. `AVLTree.tallerChild` — Return the taller of two children for a node
4. `BinarySearchTree.treeSearch` — Perform a tree search

Rebalance

The `rebalance` function is used to restructure the tree. This function is coded as a static set of transformations on the tree structure. The nodes of the tree have to change then you will have to set the left and right children and the parent of the subtrees in all cases. There are four distinct cases for structuring the tree and this has been outlined in the provided code.

The basic premise is that you need to obtain the nodes x, y, and z, and then redefine them as a, b, and c which is the order x, y, and z would appear in an in-order traversal of the tree (you hard code this and **do not** need to perform an actual in-order traversal. b then becomes the root of the new rebalanced tree a then becomes the left child of b, and c becomes the right child of b. The subtrees are reattached to the tree from left-to-right in the correct order.

The following files must be submitted to EVE:

1. *studentnumber*_p09.zip

Marksheet

1. AVLTree: tallerchild [5]
2. AVLTree: restructure [20]
3. BinarySearchTree: treeSearch [5]
4. Main: main [10]
5. Compilation and Correct execution. [10]