CSCI 3320: Data Structures
HW 4: Implementing AVL Tree
Due: 11/23 (Thu) 11:59 pm (10 points)

1. Project Details

In this project, you will implement AVL Tree. Your program will provide a simple menu (UI) for a user to manipulate the tree as follows:

0. `menu:` Shows a menu for a user to choose.
1. `insert:` Insert a new key (data) into the tree. It should not only insert a node into the tree but should make sure the tree is still a valid AVL tree. That is, this function may need to call some rotating functions. If the user chooses this menu, your program will receive a key that will be inserted to the tree from the user.
2. `contain:` Shows if the key exists in the tree. if the user chooses this menu, your program will receive a key that will be searched.
3. `height`: Shows the height of a given key if the key exists in the tree. if the user chooses this menu, your program will receive a key that will be searched and show the height only if the key exists.
4. `depth:` Shows the depth of the given key if the key exists in the tree. if the user chooses this menu, your program will receive a key that will be searched and show the depth only if the key exists.
5. `min:` Shows the minimum key in the tree.
6. `max:` Shows the maximum key in the tree.
7. `print:` Shows the tree. The code will be provided.
8. `Exit:` Terminates the program.

- Program execution example

---------- AVL Tree ----------
0. Show menu
1. Insert a new key
2. Check if a key exists
3. Find the node's height
4. Find the node's depth
5. Find the min value
6. Find the max value
7. Print tree
8. Exit

> **1**

input a key: **5**

5 is added to the tree

> **1**
input a key: **10**
10 is added to the tree
> **2**
input a key: **5**
5 exists in the tree
> **2**
input a key: **4**
4 does not exist in the tree
> **3**
input a key: **5**
5's height is 1
> **3**
input a key: **4**
4 does not exist in the tree
> **4**
input a key: **10**
10's depth is 1
> **5**
5 is the minimum value in the tree
> **6**
10 is the maximum value in the tree
> **7**
5
 \
 10
…

Please note that the red colored numbers in the example indicate the user's inputs. You can assume that your program does not support a delete operation.

2. Implementation Details

For Java: A skeleton code files (**AvlTree.java and AvlNode.java**) are provided. You will need to implement the functions in **AvlTree.java** as directed in the code and you don't need to change the **AvlNode.java**. Along with these files, another source code file (**BTreePrinter.java**) will be provided for you to print the tree.

For Python: A skeleton code file (**AvlTree.py**) is provided. You will need to implement the functions in **AvlTree.py** as directed in the code.

A binary search tree code will be provided for you to start implementing AVL tree functions. The `printTree()` function is provided in the skeleton code and you just need to call it to print the tree using the root node as a parameter. **You must use (AVL) tree-related code in the slides (and given BST code) to complete this project**. Please test your program enough to handle and avoid any exceptions. **You will lose points if your program is terminated due to any exceptions.** For example, if a user puts 9 (which is not available) to the menu, you will need to handle it by showing

appropriate error messages.

Please add comments in your code for the instructor to understand your code. You can discuss this project with your classmates to get some hints. However, **it is not allowed to 1) share the solution with your classmates and 2) post this project to web sites e.g., chegg.com to get solutions.** In short, **you must complete this project by yourself**. The final exam may ask about your project implementation details.

You will get extra credit if you implement a delete function. Please make the UI menu look like below if you implement it.

---------- AVL Tree ----------

…

7. Print tree
8. **Delete a key**
9. Exit

3. Deliverables

Please submit your code and report for performance measurement on Canvas by the due date. Please put your information in the code.

4. Grading

The following factors will be considered while grading your submission:

Code Correctness: 70%

Coding and style (indentations, readability of code, comments in code): 10%

Exception handling: 20%