IT-309 - Assignment 5 – Binary Search Trees – Build and Operate

Assignment Given: 10/20/2020
Assignment Due: 11/03/2020, 11:59 pm

In this assignment you are asked to create a Python program in a Jupyter notebook that reads a transaction stream that builds and operates a Binary Search Tree (BST) based on the stream. This will involve building and operating one or more trees in succession. To do that the program reads a stream of transactions that can perform several types of operations. The transactions include a transaction code and a numeric data node value to be stored in the BST. The transaction types and meanings:

Transaction Code	Transaction Argument	Definition/Action to be taken
I	Integer value	Insert a node into the BST (or establish the BST if it's empty) – return True if inserted, otherwise False (e.g. if the node value is already in the tree)
F	Integer search value	Find a node with a given value in the tree, return the value by displaying it, otherwise throw an exception
R	Integer value	Remove the value from the BST if found – return the value if found and removed, otherwise throw an exception
D	<none>	Display (print) the entire tree in a given format and the number of nodes and height at the time of display
E	<none>	"Erase" the current tree by instantiating a new tree object for the next transactions

In creating and coding the program:

1. The input is a single transaction text file, with the following description:
   - contains a series of transactions, one per line
   - no limit on the number of transaction lines, but plan on no more than 150
   - each transaction has the transaction code, a comma, and an integer value between 0 and 999 (max 3 digits) – ignore all other input on the line
   - the transaction stream drives what the program does – insert, find, remove, etc., and an "E" means reset the BST to being empty and start over with new transactions (if any)
   - continue processing the transaction stream until there are no more
2. the submitted code must prompt the user for the name of the transaction input file as an argument to the program
3. Examples of a transaction stream (your program should ignore comments):

```
I, 100      # establish a tree with root node value = 100
I, 162      # insert node with value = 162, return 162 (or throw exception if already there)
I, 31       # insert 31
I, 48       # insert 48
I, 92       # insert 92
I, 119      # insert 119
F, 162      # Find node = 162, return 162 (or throw exception if not found)
F, 131      # Find node = 119
R, 132      # Remove node = 132, return the value if found and removed, otherwise exception
R, 48       # Remove node = 48
D           # Display/print the tree – parentheses or other format
```

E # Reset the tree to being empty – instantiate a new tree object
(…more transactions could follow, possibly building, displaying, and resetting multiple BSTs.)

4. You must create and use a BST implementation using a linked-type (not array) strategy.
5. Implementations were presented in class and in the book. I posted some starter code to the Blackboard Jupyter notebook area. Feel free to use the starter code or that in the textbook. The book's implementation is more elaborate than mine and uses a fairly deep inheritance hierarchy. Provide a citation in your comments if using the book's code.
6. **Please note that the starter code I posted does NOT include a method to delete a tree node. You must provide that method.** However, for this assignment you do not have to balance the tree after insertions or deletions.
7. I suggest testing and debugging the code using small transaction streams, then build up to larger, more complex ones
8. Output should include an echo of the transactions as you read them ("insert node = 100" for the first line above, for example), the result of the transaction ("162 inserted"), trees displayed via the "D" transaction, and an "end of processing line" after the last transaction is read and processed. Before the end of line processing print/display the total number of trees built and total number displayed.
9. The output must also include a rendering of each final tree as it stands after a "D" transaction and when the end of the transaction stream for each tree is reached. You may use the parenthesize function I provided that displays a tree using nested parentheses. That's probably the easiest way to do it. However, you may also use visualization software to display the tree should you want to go to the extra effort.
10. I will provide at least one test transaction file for you to use in development and testing. The final test by the GTA may involve more than my test file, so satisfy the assignment specification, not my test data.

What and where to submit:

The program is to be submitted through Blackboard as a Jupyter notebook. Include the BinarySearchTree class (mine or the textbook version), the node deletion method, one or more Markdown cells at the start to identify the program and you, and additional Markdown cells as needed. No additional written report is needed.

How the assignment will be assessed

The Jupyter notebook will be visually inspected and executed for the various provided expressions. The GTA may run them against additional expressions.

Item	Assessment Description	Max Value
Python Code in a Jupyter notebook	All code submitted	5
Insertions and searches	Insertion and search ('F') transactions accurately execute	10
Deletions	Your provided deletion (node removal) method accurately removes requested nodes and leaves the tree in a valid BST state	20
Display	The tree is accurately displayed at the end of the transaction run	5
Total		**40**