CS 304 Homework Assignment 5

Due: 11:59pm, Thursday, April 9th

This assignment is scored out of 65. It consists of 4 questions. The first 3 questions are to be completed on D2L as an online quiz. There is a programming question and you will need to put all your Java programs (`*.java`) as well as output files for this question in the folder named LastName_FirstName_CS304_HW5. Zip this folder, and submit it as one file to Desire2Learn. Do not hand in any printouts. Triple check your assignment before you submit. **If you submit multiple times, only your latest version will be graded and its timestamp will be used to determine whether a late penalty should be applied.**

Short Answers

Complete the quiz for this homework on D2L by the due date. You might see there is a time limit of 120 minutes on this quiz but it is not enforced so you can ignore it. Before you complete all questions, DO NOT submit! Doing so will prevent any further changes to the answers. You can save your answers for as many times as you want before submission.

Programming Questions

P4. (41pts)

a. Completing the `BST` class

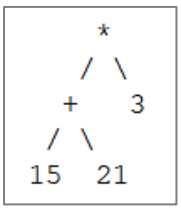
You are provided with the files "**`BSTNode.java`**" and "**`BST.java`**". You are required to complete the methods in the latter file to implement a binary search tree. This tree is designed to store only **`String`**s and it does not allow duplicate elements. You need to write the following three methods:

`add(String v)` – This **non-recursive** method takes a **`String`** and inserts it into the binary search tree, keeping the tree ordered (You can use the **`compareTo`** method to make comparisons). It does not allow inserting the same element for more than once.

`inOrder()` – This **non-recursive** method builds and returns a string that represents the in-order traversal of the binary search tree.

`min()` – This method returns the smallest element in the binary search tree. You are not allowed to create any additional structures, including but not limited to arrays, stacks, queues, or other trees.

`evaluate()` – When provided with an expression tree, this method evaluates the expression represented by the tree and returns the result. For example, it will return 108 if the root of the tree on the right is passed into the method. This is because $(15 + 21) * 3 = 108$.



Note that you are only supposed to touch the above methods. You are NOT allowed to create any other methods, instance variables, or make any changes to methods other

than these methods or files other than "`BST.java`". Points will be taken off if you fail to follow this rule.

b. Code Testing

You are provided with a test driver implemented by "**`TestBST.java`**" (**Do not make any changes to this file!**) so there is no need to write your own.

Once you have completed the methods, you can run the test. You should create a plain text file named "**`output.txt`**", copy and paste the output (if your code crashes or does not compile, copy and paste the error messages) to this file and save it.

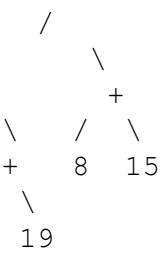
Grading Rubrics:

Code does not compile: -10
Code compiles but crashes when executed: -5
Changes were made to things other than the required methods: -5
`add` was implemented in a recursive way: -9
`inOrder` was implemented in a recursive way: -9
`min` used additional structures: -9
Has output file: 5
Code passes 12 test cases: 36 (each test case worth 3 points)

Sample Output:

```
Test 1: inOrder() ==> [Passed]
 Expected: 0 1 2 3 5 6 7 9
 Yours: 0 1 2 3 5 6 7 9

Test 2: size() ==> [Passed]
 Expected: 8
 Yours: 8

Test 3: min() ==> [Passed]
 Expected: 0
 Yours: 0

Test 4: inOrder() ==> [Passed]
 Expected: ab ac ae bc bp ck de dg eh
 Yours: ab ac ae bc bp ck de dg eh

Test 5: size() ==> [Passed]
 Expected: 9
 Yours: 9

...
```

Test 12: evaluate(root), where root points to "/" in the following tree ==> [Passed]



Expected: -22
Yours: -22

Total test cases: 12
Correct: 12
Wrong: 0