CS2613: Programming Languages Laboratory
Racket: Question 3

Overview:

The following question must be completed in Racket. Submissions must be made to GitHub. Late submissions will not be accepted. All online resources are available including official documentation, forums, videos, etc. If you are struggling with a concept and online research is not helping, then you should ask the course professor.

Learning Goals:

- Object Oriented Programming
- Binary Search Trees
- Strings

Special Node from Instructor:

I have a feeling that this will be the most challenging programming question in the course. I recommend leaving it until later in the course. Lab #19 will cover OOP in Racket and will give you examples. Remember test often to ensure that methods are working how you expect them to. Break the problem down as much as you can. Make sure to reach out if you have questions.

Question: Letter-Based Binary Search Tree

Create a BinarySearchTree class that holds a pointer to a LetterNode object (the head of the tree). Each LetterNode should hold references to a left and right letter node, a letter (as a string is likely easiest), and a frequency.

The BinarySearchTree class should have an insert and a print (in-order) method.

Write a function that takes in a word (a string) and cycles through to add every letter to the binary search tree. Nodes should be inserted in by the alphabetical order of their letter. If the letter is already stored in a Node in the BST, increment the frequency counter by one. You may assume that the word will not contain any special characters, only letters; however, it may be in upper or lower case. You should treat upper- and lower-case letters as the same letter (‘A’ == ‘a’).

You may need various helper methods throughout. Likely easiest to just have them set as public for this programming question.

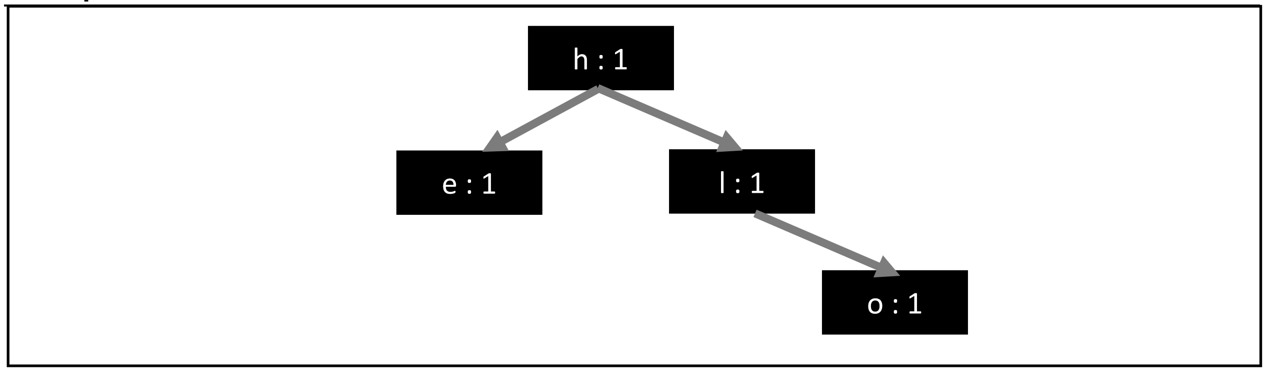
Example Code:

```
(define tree (make-object BST% null))
(word->tree "HelLloOOOO" tree)
(send tree printTree)
```

Example Output:

```
e: 1
h: 1
l: 3
o: 5
```

Example Tree:



Hints:

Here are the methods for the classes I created.

BST Methods

- Constructor(LetterNode)
  - Set the letter node to null when creating your BST.
- insert(LetterNode)
- insert-help(LetterNode, LetterNode)
- get-head()
- print-tree()
  - Calls a print method in the LetterNode class.

LetterNode Methods

- Constructor(LetterNode, LetterNode, string, number)
  - Sets values to default. Takes null (left), null (right), letter, and frequency (usually set to 1) when first creating a new node.
- increment-frequency()
- print()
  - Recursive printing here. Could be done in the BST class instead.
- + Accessors and Mutators for each instance variable.