CS 212: Object Oriented Data Structure
Project Assignment 04: Binary Trees
Assigned: Wednesday April 11, 2017
Due: Wednesday May 3, 2017 (at time of final exam – on Moodle and paper submission)

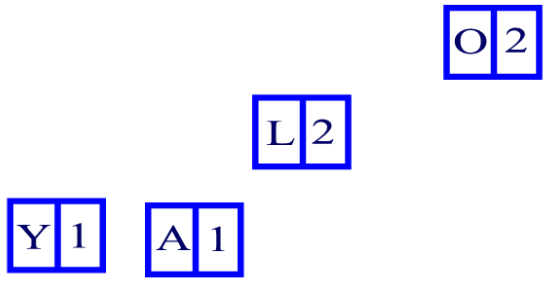
Purpose: Use a Binary Tree to convert a word or phrase to binary.

Problem Description: Write a Java program that supports converting a word or phrase to a binary string and test your code.

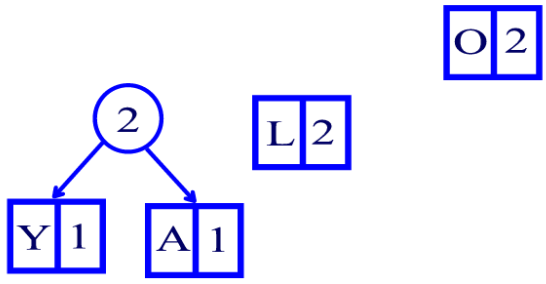
- Ask the user for a word or phrase. Example: LOYOLA
- Create an ArrayList of Data objects for that word or phrase. Example: L:2, O:2, Y:1, A:1. You may need some helper methods to check if a character has already been processed (for example the second L and the second O) or is a new character (the first L, the first O, Y, and A). A Data object is an object of the Data class (which you need to code): a Data object is made up of a char (the letter) and an int (the frequency of that letter).
- Test/debugging: Output the contents of the ArrayList
- Order the Data objects in that ArrayList in ascending order based on frequency (you can unload the ArrayList into an array, sort the array, and unload back the array into the ArrayList). Example: Y:1, A:1, L:2, O:2
- Test/debugging: Output the contents of the sorted ArrayList
- Using that ArrayList, create a forest of Binary Trees; that forest can be stored in an ArrayList of BinaryTrees; each Binary Tree only contains one Node.
- With that forest of BinaryTrees, loop in order to build the resulting Binary Tree as follows:
    - At each iteration of the loop, the 2 Binary Trees with the smallest "frequencies" are combined to make one Binary Tree.
    - The 2 Binary Trees that were combined are eliminated from the forest, whereas the resulting Binary Tree is added to the forest; thus, at each iteration of the loop, the forest contains one fewer tree. Eventually, it will contain only 1 Binary Tree.
    - The "frequency" of that resulting Binary Tree is the sum of the frequencies of the 2 Binary Trees.
    - The resulting Binary Tree is added to the forest in a way that keeps the forest sorted in ascending order OR you should sort the forest after adding the resulting Binary Tree. Note: if you prefer, instead of an ArrayList of BinaryTrees, you can use a sorted linked list of Binary Trees.
    - Test/debugging: At each iteration of the loop, output the contents of the forest of Binary Trees (i.e. the data stored in the root of each binary tree).
- The resulting Binary Tree contains the codes for each letter. Each letter is stored in the leaves. Traverse the tree, for example using a preorder traversal, in order to assign the binary code to each letter. For this, you can use a helper class that includes a char (the letter) and a String

(the binary encoding) instance variables and build an ArrayList of these objects as you traverse the Binary Tree. That ArrayList can be an instance variable of the BinaryTree class.

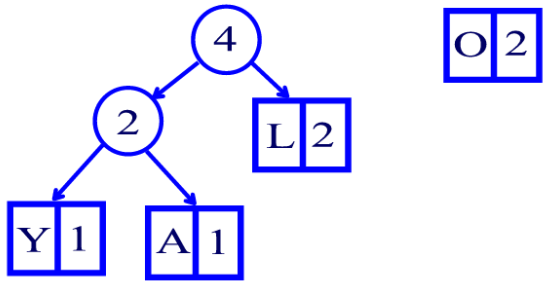
- Test/debugging: Output the contents of that ArrayList (a list of char/String pairs).
- Output the encoding the original word as a binary string.



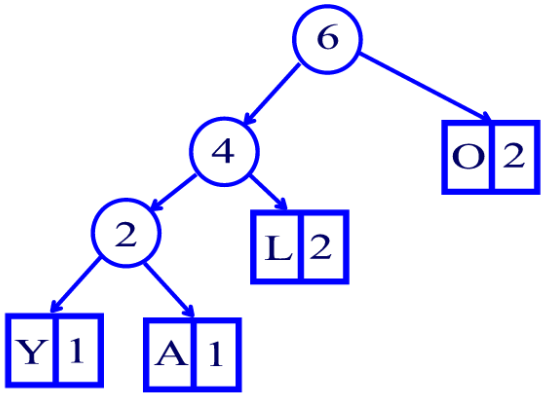
The forest of binary trees for LOYOLA



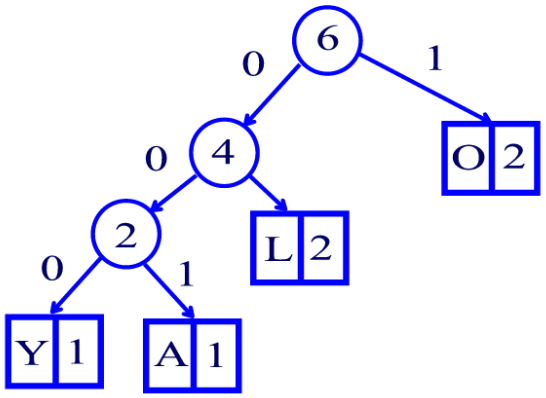
Building the binary tree for LOYOLA



Building the binary tree for LOYOLA



The resulting binary tree for LOYOLA



The encoding scheme for LOYOLA

Y = 000
A = 001
L = 01
O = 1

LOYOLA = 011000101001

Implementation. Included classes:

Node class: Similar to the Node class we used in class for Binary Trees, but storing user-defined data (defined below – the class with a char and an int).

User-defined class (Data - you choose the name of that class): Includes 2 instance variables:

- A char, to store a letter
- An int, to store the frequency of that letter in the word

Another user-defined class (Code - you choose the name of that class): Includes 2 instance variables:

- A char, to store a letter
- A String, the representation of that letter with 0s and 1s

BinaryTree class: includes the following instance variables and methods:

Instance variables:

- root, a Node
- frequency, an int
- An ArrayList of Codes (see class above)

Methods:

- A method to sort an array of BinaryTrees based on frequency (in ascending order).
- A method to combine two Binary Trees into one.
- A method to accept a forest of Binary Trees as its parameter and build a Binary Tree from it.
- Regular Traversal methods (preorder, postorder, inorder).
- Traversal to build the codes.
- A method to output a String representation of the codes.
- Helper methods as necessary.

Client Encoding class: Asks the user for a word and encode it