COSC 3333 - Data Structures & Algorithms II **Spring 2022**
Programming Assignment 3 **Due Date: Wednesday, March 9 @ 11:59 p.m**

Download the code provided by the textbook in order to complete this program.
This is a two part assignment.

Part I:

Write a program that reads a word (series of ALL CAPITAL LETTERS) from the keyboard and builds a **Binary Search Tree** out of them. You can use the code in the book to build this tree, but you must incorporate the following additional rules for this assignment.

To start you may want to modify the class `Node` in the book to have a `parent` attribute in addition to the left child and the right child. This attribute will help with the rules of this assignment.

Although you must follow the BST rules to build a valid BST out of these uppercase letters, but you must make sure that:

1. *No vowel parent can have a vowel as a child unless the parent's sibling is also a vowel.* This means that you can insert a vowel as a child of another vowel only if its uncle-to-be is also a vowel.

2. *No consonant parent can have a duplicate value as a child i.e. the parent and a child cannot be the same consonant letters. This duplicate parent-child rule only applies to consonants and not vowels. For example letter A can be the child of parent-A as long as the child-A has a vowel uncle.* Note: We agreed to insert the legal duplicates as the left child.

3. When you encounter a letter that could not be inserted based on the above rules, you will save them in a proper data structure to be re-inserted following the rules of part II of this assignment, and move on to the next letter. When you are done populating the BST with the letters that you could properly insert, you will re-insert the letters that were put on hold IN THE EXACT ORDER they were put on hold and in the tree based on the rules of Part II. This should give you a hint about what data structure you need to use for these letters for further processing.

4. **BONUS:** After the BST is properly populated, you must visit each node and compute its height. You need to modify the book's code for class `Node` to incorporate this integer attribute and populate it as you visit each node. You MUST write a method to compute all nodes' height and assign them to their `height` attribute.
   Why do you need to wait until the tree is fully populated to start computing each node's height and could not decide on the height as you were inserting the nodes?
   I leave it up to you to decide whether to write this method recursively or iteratively.

The following is the list of what is expected to output after completing part I:

1. Display the BST (using the book's method)

2. Display the contents of the data structure you used to store the letters that could not be inserted (if any).

3. **BONUS:** Display the information of every node in the BST: the letter followed by their height; one node per line. This part along with the method to compute each node's height will possibly earn you **15 bonus points.**

Part II:

By the time you get to this part, you have your string of uppercase letters configured as a BST following the rules for Part I and potentially have a data structure that contains the letters that could not be inserted in BST and now you get a chance to re-insert them based on a different set of rules.
In case you have letters left from Part I, follow the methodology we covered in class and re-insert them IN THE EXACT ORDER THEY WERE PUT ON HOLD, and in the first available spot you find in the BST, starting from the left side.

When done, display the tree using the textbook's display method, to show the letters that were inserted based on Part II's rules.

- *The program must be written in Java.*
- *You must upload all relevant .java files, including the ones from the book, on blackboard to ensure successful execution of your code, by the due date.*
- *The submission that is missing required files/classes will be considered void.*
- *Programs that do not compile will be considered void.*
- *This is an individual assignment. If students submit fully or partially identical programs, all individuals involved will earn a grade of zero.*