Efficient encoding for text

Texts can be encoded efficiently, if common characters are encoded into shorter bit sequence, rare characters into longer bit sequence. This can be done with an encoding tree which is a binary tree.

**Definition 1. weight of encoding tree**

*If the root node is a leaf: the count of the character is used in key text. Otherwise: sum of the weight of two subtrees.*

We can generate an encoding tree with key text by following procedure.

1. Sort characters in key text with their count used in descending order.

2. Pick two rarest character and make each of those into binary trees having one node.

3. Construct initial encoding tree by combining two binary trees into one which is having heavier subtree as its left branch.

4. Iterate below until all characters are included in encoding tree.
   
   (a) Pick the rarest character not included in encoding tree and make it into a binary tree with one node.
   
   (b) Construct new encoding tree with new binary tree in step (a) and existing encoding tree, which is having heavier subtree as its left branch. (If two subtrees have same weight, take existing encoding tree as right subtree.)
Encoding with encoding tree can be done by following procedure.

1. Iterate below procedure for every character in text to encode.

   (a) Get the path from root th the node for current character in the encoding tree.
   (b) Encode the path into a bit sequence.
      i. Left branch : 0
      ii. Right branch : 1

2. Concatenate bit sequences.

Example

Key Text : "teaatt"
Encoding tree : ( (t:3) ( (a:2) (e:1) ) )
Encoded bit sequenc for "tee"
t: L → 0, e: RR → 11, e: RR → 11 ⇒ "01111"

Problems

1. Implement binary tree which meets following requirement.

   (a) Class signature : public class BinaryTreeNode
   (b) Needed methods
      public BinaryTreeNode getLeftBranch()
      getter for left subtree. If none, return null.
      public BinaryTreeNode getRightBranch()
      getter for right subtree. If none, return null.
      public void setLeftBranch(BinaryTreeNode newSubtree)
      setter for left subtree.
      public void setRightBranch(BinaryTreeNode newSubtree)
      setter for right subtree.
      public int getWeight()
      returns weight for entire tree. If the node is a leaf, return the count of the character is used in the key text.
public String getCharacter()
    If the node is a leaf, returns the character the node represents. Otherwise return null.

2. Implement class "EfficientEncoding" which meets following requirements.

   (a) Class signature : public class EfficientEncoding
   (b) Needed methods

         public String encode(String textToEncode)
                   returns String representing the bit sequence encoded from textToEncode.
         public String decode(String encodedBits)
                   returns String representing the text decoded from bit sequence.

3. Implement class "EncoderGenerator" which meets following requirements.

   (a) Class signature : public class EncoderGenerator
   (b) Needed methods

         public static EfficientEncoding generateEncoding(String keyText)
                   returns an instance of EfficientEncoding which has encoding tree generated from key text.

4. Define a class named "Main" and test your implementation of "EfficientEncoding".

1Please use following package name → kr.ac.postech.csed233.hw3.YOUR_STUDENT_NO