2018-1 Computer Programming Assignment 3

You should do the assignment on your own. You are not allowed to share codes with others and/or copy codes from other resources. If you get caught, you will lose all points from this assignment.

Grading will be done in Linux environment using Java (OpenJDK) 8, identical to that inside the lab machines. Keep that in mind when writing code in other environments. Programs having any kinds of compile errors will receive neither compile nor test case points.

Do not change the format of input and output. You cannot get any points if you do not follow the output specification and print any sorts of different output from the one in this PDF file.

Write everything, including comments, in English.

Upload your work in ETL. You should submit only a single TAR or ZIP file containing those files:

[1] Problem 1: CBTree.java (C/B/T in upper case!).
[2] Problem 2: LLString.java (L/S in upper case!).

Do not include any subdirectories or any other files inside your archive, so that we can see your source codes right after unzipping it. Graders will deduct some of your grade if you do not follow this: note that you are not going to get any grades regarding this issue sometime later.

Due of this assignment is 11PM on May 17th. No late submission is allowed.

If you have any questions, write an article to the Class Q&A board in ETL so that everyone can see what is going on. TAs will try to respond your questions and announce modifications of the specification promptly, if any. However, TAs will not be able to answer questions after 5PM on May 15th: it is kindly recommended to start your work as early as possible.

Problem 1

Write a program CBTree.java that implements a class `CBTree`. This class constructs a binary tree satisfying the properties below.

<Description>

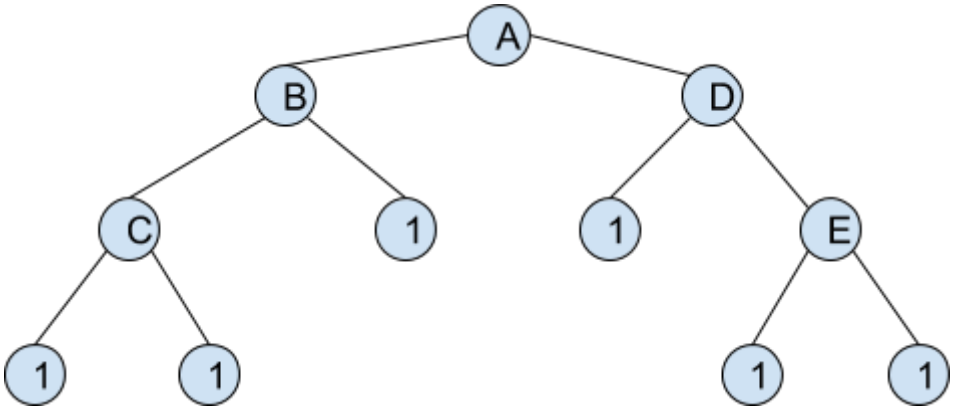
- A class `CBNode` is given to you, which defines the two `char` field variables `label` and `character`.
  - `label` stores either 0 or 1, based on `st` mentioned below.
  - `character` stores a single upper-case letter.

  You are not allowed to modify `CBNode`.
- Constructor of class `CBTree` is given as: `CBTree(String st, String con)`.
  - `st` is a sequence of 0 and 1, representing the tree in preorder. 0 means a node has two children, whereas 1 denotes a leaf node.
  - `con` is a sequence of upper-case alphabets, with each corresponding to its relative internal nodes in preorder. Length of `con` is equal to number of 0s in `st`.

  For example, if we write a statement:

  ```
  CBTree cbt = new CBTree("00011101011", "ABCDE");
  ```

  this denotes a tree:



- `CBTree` class contains the following methods:
  - `String postOrderTraversal()` returns a `String` obtained by concatenating the `label`s of `CBNode`s in postorder. For instance, invoking this method to the tree above returns CBEDA.
  - `String inOrderTraversal()` returns a `String` obtained by concatenating the `label`s of `CBNode`s in inorder. For instance, invoking this method to the tree above returns CBADE.
  - `String postOrderStructure()` returns a `String` obtained by concatenating a `String bit` per `CBNode`, traversed in postorder. `bit` is represented by a character: it is 0 if a certain node has two children, and 1 otherwise. For instance, invoking this method to the tree above returns 11010111000.
  - `String inOrderStructure()` returns a `String` obtained by concatenating a `String bit` per `CBNode`, traversed in inorder. For instance, invoking this method to the tree above returns 10101010101.
- Do not add any other field variables in `CBTree`. However, you may add other methods than the four methods.
- Do not import any classes. If you want to use any helper structures (e.g., FIFO (First-In-First-Out) structures), implement on your own.

<Example> /* Note: graders will execute the code similar to this. */

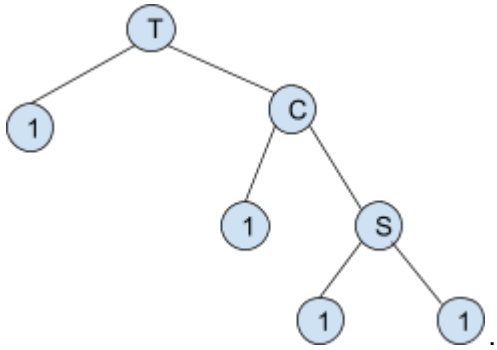
[cp00@cp] ~$ `ls`

CBNode.java CBTree.java CBTreeTest.java

[cp00@cp] ~$ `cat CBTreeTest.java`

```
public class CBTreeTest
{
	public static void main(String[] ar)
	{
		CBTree cbt = new CBTree("0101011", "TCS");
		System.out.println(cbt.postOrderTraversal());
		System.out.println(cbt.inOrderTraversal());
		System.out.println(cbt.postOrderStructure());
		System.out.println(cbt.inOrderStructure());
	}
}
```

/* Note: `cbt` refers to:



. */

[cp00@cp] ~$ `javac CBTreeTest.java`

[cp00@cp] ~$ `java CBTreeTest`

SCT

TCS

1111000

1010101

[cp00@cp] ~$

Problem 2

Write a program LLString.java that implements a class `LLString`. This class constructs a sequence of letters represented as a linked list, satisfying the properties below.

<Description>

- You need to implement those operations working in `LLString`. Refer to the API document to get the details.
  - `LLString(String str)`: this is a constructor, generating a `LLString` with its content being `str`.
  - `char charAt(int index).`
  - `int compareTo(String anotherString),`
    `int compareTo(LLString anotherLLString).`
  - `int compareToIgnoreCase(String str),`
    `int compareToIgnoreCase(LLString llstr).`
  - `LLString concat(String str),`
    `LLString concat(LLString llstr).`
  - `int indexOf(int ch),`
    `int indexOf(int ch, int fromIndex),`
    `int indexOf(String str),`
    `int indexOf(String str, int fromIndex).`
  - `int length().`
  - `LLString replace(char oldChar, char newChar).`
  - `LLString substring(int beginIndex),`
    `LLString substring(int beginIndex, int endIndex),`
  - `String toString().`
- It is strongly discouraged to use original methods from `java.lang.String`.
- Simple copying and pasting `String`s is not allowed.
- Do not import any classes.

<Example> /* Note: graders will execute the code similar to this. */

```
[cp00@cp] ~$ ls
LLString.java LLStringTest.java
[cp00@cp] ~$ cat LLStringTest.java
public class LLStringTest
{
	public static void main(String[] ar)
	{
		LLString llstr = new LLString("Programming");
		System.out.println(llstr.charAt(2));
		System.out.println(llstr.length());
		System.out.println(llstr.toString());
	}
}
/* Note: estimate the result of other operations that are not dealt in this example on your own. */
[cp00@cp] ~$ javac LLStringTest.java
[cp00@cp] ~$ java LLStringTest
o
11
Programming
[cp00@cp] ~$
```