Computer Programming Assignment 4

Due date: 2017-12-1 23:59:00

Checkpoints

1. You should do the assignment in your own. You are not allowed to share code with others and/or copy code from other resources. If you are caught, as in the syllabus, you will get a failing grade.
2. Grading will be done in Linux environment using java 1.8.
3. Program failed to compile/run will result 0.
4. Do not infinite loop your program to repeat unless you are told so.
5. Do not change input/output format unless you are told so.
6. Do not color console.
7. Write your name and student number at top of program as a comment

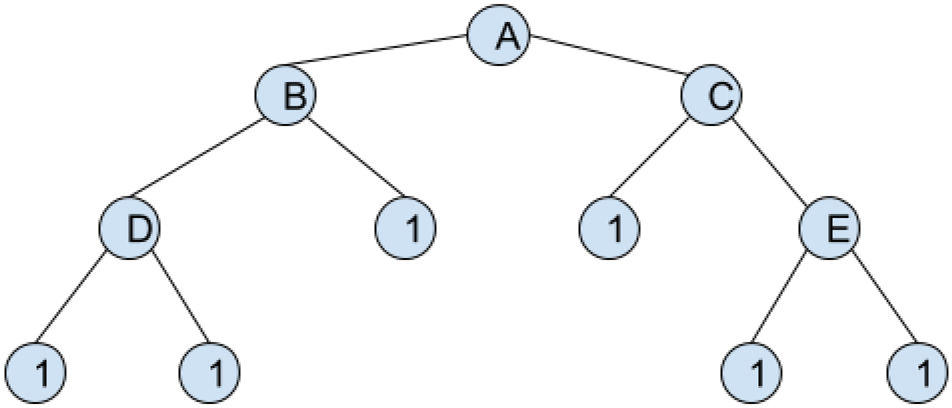
Submission

1. Submit your assignment on eTL.
2. Zip your file (or tar) as ‘<Student ID>-assign4.zip’
    a. Do not create subfolder
    b. Do not include package
3. No late submission is allowed

Binary Tree

- Write a class Assignment4_1.java
  - Constructor takes two strings, and your job is to build tree
    - First string consist only ‘0’ and ‘1’ representing level order.
      - ‘0’ means node has two children
      - ‘1’ means node is leaf node
    - Second string consists of a sequence of characters, each corresponding to each of the internal nodes, in level order.
      - Length of second string is equal to number of ‘0’s in the first string
      - 1st character in second string corresponds to the 1st ‘0’ in first string, 2nd character corresponds to the 2nd ‘0’ in the first string, and so on.
- Do not create any additional field variables in the class Assignment4_1.java.
- Do not import any classes. If you wish to import FIFO data structure, write your own. (Assignment4_2 consists linked list. If you are implement FIFO and use separate classes, make sure not to put any overlap that may cause compile error.)
- Make sure you store information on Node4_1 class according to Tester4_1.java
- Submit only Assignment4_1.java file (and your FIFO). Any modification to Node4_1.java will be overridden.

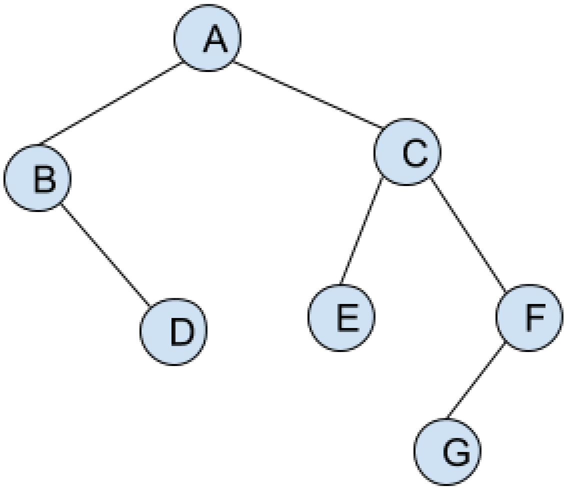
```
new Assignment4_1("00001101111", "ABCDE");
//Node4_1.type is 0 for node with labels
```



Binary Tree2

- Write a class Assignment4_2.java
    - Constructor takes two strings S1 and S2, and your job is to build tree
        - String S1 is a bit sequence.
            - Each node x is represented by two bits x_1 and x_2: x_1 = 1 if x has a left child, and 0 otherwise. Similarly, x_2 = 1 if x has a right child and 0 otherwise.
            - Sequence of bits are given in the preorder traversal of the tree nodes..
        - Second string consists of a sequence of characters, each corresponding to the label of a node in the tree. This sequence is obtained by traversing the tree in level order and writing down the labels of the nodes.
            - Length of S2 is half of length of S1
    - **Construct tree in recursive**
        - You may construct tree with S1 first then fill S2 after.
    - String report_bits_preorder() returns string obtained by concatenating the “bits” of the nodes in preorder, where the “bits” of a node x are the two bits x_1 and x_2, as defined above.
    - String report_bits_levelorder() returns string obtained by concatenating the “bits” of the nodes in levelorder, where the “bits” of a node x are the two bits x_1 and x_2, as defined above.
    - String report_preorder() returns string obtained by concatenating the labels of the nodes in preorder.
    - String report_levelorder() returns string obtained by concatenating the labels of the nodes in levelorder.
- Do not create any additional field variables in the class Assignment4_2.java.
- Submit only Assignment4_2.java file (and your FIFO). Any modification to Node4_1.java will be overridden.

```
new Assignment4_2("11010011001000", "ABCDEFG");
```



LinkedList : String

In Java's String class, useful methods are provided for easier handling or conversion. String class stores an array of 'char'. How about implementing String with LinkedList?

In the problem, java interface file will be provided as a reference to necessary methods to be implemented. You should implements the java interface, and just implement methods here, that means you do not have to write main method.
Do not use java's String methods to implement the following methods, except "toString()", "toCharArray()", or "charAt(int index)"

- void remove(String subs), void remove(LinkedString subs)
  - This method should remove the sequent 'subs' from original char linked list. When you remove 'subs', you must check char sequence again to check if 'subs' appears in the string after removing of 'subs'
  - e.g.) String: abbcccc , subs : bcc => step 1: abcc => step 2: a
    result: a
- String toString()
  - Convert LinkedList String to original String.
- char charAt(int index), int length()
  - charAt(int index) returns the char at 'index'.
  - length() just returns the total length of characters in the LinkedString.
- LinkedString substring(int startIndex, int endIndex)
  - It returns a substring starting from and containing char at 'startIndex' ending in and not containing char at 'endIndex'
- boolean contains(LinkedString substr), boolean contains(String substr)
  Returns true if 'substr' is a substring of this string.
- int compareTo(LinkedString/String str), int compareToIgnoreCase(LinkedString/String str)
  - Returns 0 if all characters in this string is the same with 'str', positive number if this string is lexicographically larger and negative number otherwise.
  - compareToIgnoreCase method ignores cases of alphabets such as 'a' and 'A'.
- LinkedString concat(LinkedString str) , LinkedString concat(String str)
  - Concatenate 'str' at the end of this string, and return the concatenated string.
- int indexOf(char ch), int indexOf(String str)
  - "indexOf" returns the leftmost index where 'ch' or 'str' occurs.
- LinkedString replace(char oldChar, char newChar)
  - Replace all 'oldChar' characters with 'newChar' and returns this string.

Submission

- Make a folder that contains all of your java files named as “Assignment4_3”, and contain this file in the your assignment 3 zip file.