COMP 2011, Assignment #3 **DUE ON: Nov. 18, 2016 12:30pm**

Notice

- (**NEW**) submission procedure (Any inquiry about this should be addressed to Mr. WU Hanqing):

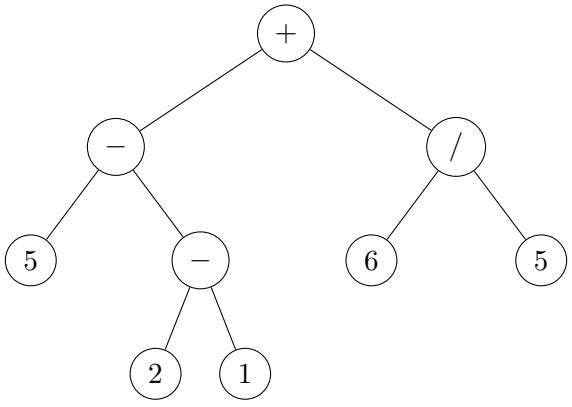
  1) Create a folder and name as `<student_no>_<yourname>` E.g., 12345678d_CHANTaiMan
  2) Name the .java file as `A3_Q1_<student_no>_<yourname>_<class_name>.java` E.g., `A3_Q1_12345678d_CHANTaiMan_Sorting.java` (A for assignment, P for project) **But please don't change the class name in the Java file.**
  2) Put all your .java files into this folder, includingly java files and screenshots. Please DO NOT submit the entire project.
  3) Compress this folder (all compressed format are acceptable, e.g., .jar, .rar, .zip, .7z) and submit the compressed file to the Blackboard.
  4) Any wrong file naming and submission will result in mark deduction.

- All other requirements of assignment#2 also apply for this one.

1. (20 points) Write the merge sort for linked list. You must use the given class `LinkedList.java`. Your method should run in $O(n \log n)$ time, where the n is the number of nodes in the list, and you can use only $O(1)$ extra spaces.

2. (20 points) Finish the hoare method in Lab 9.

3. An arithmetic expression can be represented with a binary tree: Here we consider only binary operations, so this tree is special in the sense that if a node has either zero or two children. Note that this is NOT a binary search tree. For example, the tree for the expression
   `"5 2 1 - - 6 5 / +"` is



   (a) (30 points) Given a postfix expression, build a binary tree (`Postfix.buildTree`). *Hint: (0) You can use.`comp2011.lec4.Postfix` for the codes of parsing an expression as a `String`. (1) We have two different kinds of nodes, operators (`char`) and operands (`int`). How to store them? (2) For the example above, the root should be `'+'`. So the nodes should be inserted in the reversed order. How? (3) Processing the tokens from right to left, the second token `'/'` is the right child of `'+'`. Its left child `'-'` is somewhere we don't know yet. How do we keep track of this? (4) You may find stacks very useful (indispensable) for this question.*

(b) (10 points) Implement the `inorder` and `postorder` methods of the class `ExpressionTree.java` so that they display the expression in its infix and postfix formats respectively. For the example, you should print $((5-(2-1))+(6/5))$ instead of $(((5)-((2)-(1)))+((6)/(5)))$ to get full points. You can get bonus points if you print $5-(2-1)+6/5$, i.e., only print the necessary parentheses. *Hint: When displaying an arithmetic expression in infix form, we need to use parentheses to make explicit the ordering of operations. For example,* $5-1-2$ *and* $5-(1-2)$ *are different.*

(c) (20 points) Implement the `recSize` and `size` methods of the class `ExpressionTree.java` to calculate the number of nodes in the expression tree. The `size` method must not use recursion.

All your methods should run in $O(n)$ time, where n is the number of nodes in the tree. Please test your codes with more nontrivial expressions to make sure they really work.