COMP 2011, Assignment #3 **DUE ON: Nov. 26, 2015 11:59pm**

1. (10 points) Write two algorithms that find the number of leaves of a binary tree. One of your algorithms must use recursion, and the other must not. Both your algorithms should run in $O(n)$ time, where n is the number of nodes in the input tree.
   You may want to use classes in `comp2011.lec9` if you do it in Java.

2. (20 points) Write the in-place heap sort algorithm for an array of integers. You may want to adapt the methods in `comp2011.lecA.Heap.java`, and use `comp2011.lec2.SortingTester` for testing.

3. (20 points) Given a sorted doubly linked list, write an algorithm to transform it to a balanced binary search tree. Try to make your algorithm as efficient as possible.

4. (10 points (bonus)) This is **not** a programming question. This question is the most challenging one of the whole course, and needs all previous knowledge and significant efforts. You're welcome to discuss with your fellow students, but everyone has to write the proof independently. Plagiarism has consequence!

   The input to the following algorithm is adjacency lists of a graph on n vertices and m edges. It assigns to each vertex a label and a number, both nonnegative integers. Choose appropriate data structures to implement the algorithm in $O(n+m)$ time.

   Explain in details how every step is implemented, and analyze the complexity of the whole algorithm. You can get partial points if your implementation takes $\omega(n+m)$ time, provided it is correct.

```
1.   for each vertex $v$ do label[$v$] $\leftarrow 0$;
2.   for $i \leftarrow n, \ldots, 1$ do
3.       choose an unnumbered vertex $v$ with the largest label;
4.       number[$v$] $\leftarrow i$;
5.       for each unnumbered neighbor $u$ of $v$ do label[$u$]++.
```

Hint: all the following operations need to be done in $O(1)$ time: (i) finding a vertex with the largest label; (ii) finding the label of a vertex; and (iii) updating the label of a vertex.