COMP 2119A Introduction to Data Structures and Algorithms
Test 2
Date: 01 December 2016 (Thursday)
Time allowed: 100 minutes (2:30pm – 4:10pm)

[Note that for the questions involving algorithm design, before you present your algorithm, try to describe your idea first if you think that would help the marker to understand your solution! More marks will be given to faster algorithms.]

1) [25%] Consider each of the following cases. If it is correct, state that it is CORRECT (NO NEED to give a proof). If it is incorrect, give a counter-example.

(a) In a binary search tree, if node *u* has only one child, its successor, if exists, has no <u>left</u> child.
(b) In a binary search tree, if node *u* has two children, its successor has no <u>right</u> child.
(c) Given only the preorder traversal of a <u>binary search tree</u>, we can uniquely determine the topology of the tree.
(d) A constant number of rotations is sufficient to re-balance an AVL tree in the worst case when we <u>add a new node to it</u>.
(e) A constant number of rotations is sufficient to re-balance an AVL tree in the worst case when we <u>delete a node from it</u>.

2) [10%] Define an M-AVL tree (modified AVL tree) as follows. An M-AVL tree is a binary search tree such that for every node, the difference between the height of its left subtree and the height of its right subtree is at most 2. Prove or disprove that the height of an M-AVL tree with n nodes is of $O(\log n)$.

3) [15%] Insert the following numbers into an initially empty AVL tree one by one. Show the final AVL tree (you do NOT need to show the intermediate tree). Then, delete 12, then 90. Show also the resulting tree after EACH deletion.

34, 68, 90, 100, 12, 7, 9, 6, 25

[Note: You need to follow the exact procedures taught in the lectures, otherwise a separate proof is required to show that your method works correctly.]

4) (a) [10%] Given a sequence of numbers, prove or disprove that the same heap will be obtained no matter you use the top-down approach or the bottom-up approach.

(b) [10%] Insert the numbers in Q(3) into an initially empty <u>max</u>-heap one by one (i.e. top-down approach). Show the final resulting heap (NO need to show any intermediate step). Show the resulting heaps after executing the Extract-Max operation twice.

5) (a) [5%] Sort the following numbers in increasing order using insertion sort. Show the resulting list of numbers after each round.

87, 56, 21, 47, 91, 35, 66, 0, 37, 101, 25

(b) [7%] Given *n* numbers (not necessarily distinct), stored in an array *A*[1..*n*], design an algorithm to return a number with the highest frequency and analyze the time complexity of your algorithm.

(c) [8%] Repeat (b) if we know that these *n* numbers are integers from [0..*k*], where *k* is a constant.

(d) [10%] Construct the Huffman Code for the following message [ignore spaces]:

“i love you so much”