<u>CSCE 500</u> Midterm Exam #1 10/5/2016

1. Is n^2 an asymptotically-tight bound of $n^2/(\lg n)$? of $(n^{2.5})/400$? (Briefly explain. 6%)

2. The algorithm for finding the maximum subarray that crosses the midpoint of Array $A[1 \ldots n]$ includes the main routine of FIND-MAXIMUM-SUBARRAY(A, *low*, *high*), which calls FIND-MAX-CROSSING-SUBARRAY(A, *low*, *mid*, *high*), as follows. Complete the <u>six (6) missing statements</u> in FIND-MAX-CROSSING-SUBARRAY below. (12%)

<u>FIND-MAX-CROSSING-SUBARRAY(A, *low*, *mid*, *high*)</u>

// Find a maximum subarray of the form $A[i \,..\, mid]$.
left-sum $= -\infty$
sum $= 0$
for $i = mid$ **downto** *low*
    sum $=$ *sum* $+ A[i]$
    $\boxed{\qquad\qquad\qquad}$
        $\boxed{\qquad\qquad\qquad}$
        $\boxed{\qquad\qquad\qquad}$
// Find a maximum subarray of the form $A[mid+1 \,..\, j]$.
right-sum $= -\infty$
sum $= 0$
for $j = mid + 1$ **to** *high*
    sum $=$ *sum* $+ A[j]$
    $\boxed{\qquad\qquad\qquad}$
        $\boxed{\qquad\qquad\qquad}$
        $\boxed{\qquad\qquad\qquad}$
// Return the indices and the sum of the two subarrays.
return (*max-left*, *max-right*, *left-sum* + *right-sum*)

3. Derive the tight lower and upper <u>bounds</u> of the following recurrences:
$T(n) = 2\cdot T(n/4) + T(n/2) + c\cdot n$ (10%)
$T(n) = 2\cdot T(n/2) + n\cdot \lg(n)$. (8%)

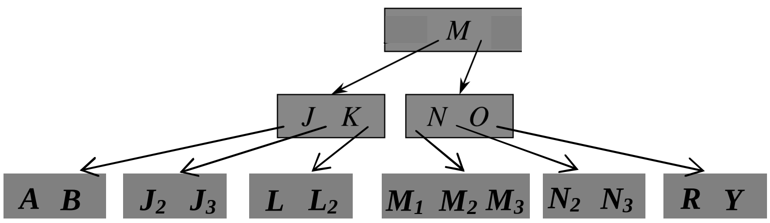
4. For any n-key <u>B-tree of height h</u> and with the minimum node degree of $t \geq 2$, <u>prove that</u> h is no larger than $\log_t \frac{n+1}{2}$. (Hint: consider the number of keys stored in each tree level.) (12%)

5. The utilization efficiency of a hash table depends heavily on its hashing function(s) employed. Describe with a <u>diagram</u> to illustrate how a <u>multiplication method</u> of hashing works on a machine with the word size of w bits for a hash table with 2^p entries, $p < w$. (10%)
Explain briefly (1) how <u>perfect hashing</u> works, and (2) how <u>Cuckoo hashing</u> works under two hash functions of h_1 and h_2. (12%)

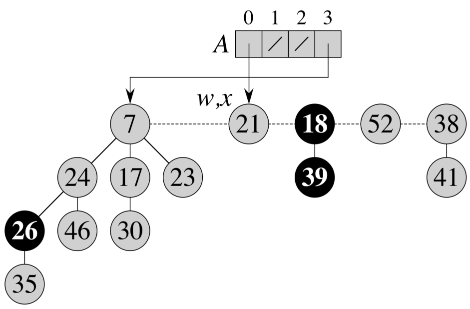
6. The <u>binary search tree</u> (T) facilitates key search and it involves several operations to maintain the tree property when a node (z) is deleted, as shown in the following pseudo code, TREE-DELETE(T, z), where TRANSPLANT(T, u, v) replaces the subtree rooted at u with one rooted at v. Fill in the last <u>three missing statements</u> in the pseudo code below. (10%)

```
TREE-DELETE(T, z)
if z.left == NIL
    TRANSPLANT(T, z, z.right)          // z has no left child
elseif z.right == NIL
    TRANSPLANT(T, z, z.left)           // z has just a left child
else // z has two children.
    y = TREE-MINIMUM(z.right)          // y is z's successor
    if y.p ≠ z
        // y lies within z's right subtree but is not the root of this subtree.
        TRANSPLANT(T, y, y.right)
        y.right = z.right
        y.right.p = y
    // Replace z by y.
    [                ]
    [            ]
    [            ]
```

7. Given the initial <u>B-tree</u> with the minimum node degree of <u>$t = 3$</u> below, show the results (a) <u>after deleting</u> two keys in order: M then R and (b) followed by <u>inserting</u> the key of L_1, with $L < L_1 < L_2$. (Show the result after each deletion and after insertion; 10%)



8. A <u>Fibonacci min-heap</u> relies on the procedure of CONSOLIDATE to <u>merge trees</u> in the root list upon the operation of extracting the minimum node. Given the following partially consolidated diagram, <u>show every subsequent consolidation step</u> till its completion. (10%)



Good Luck!