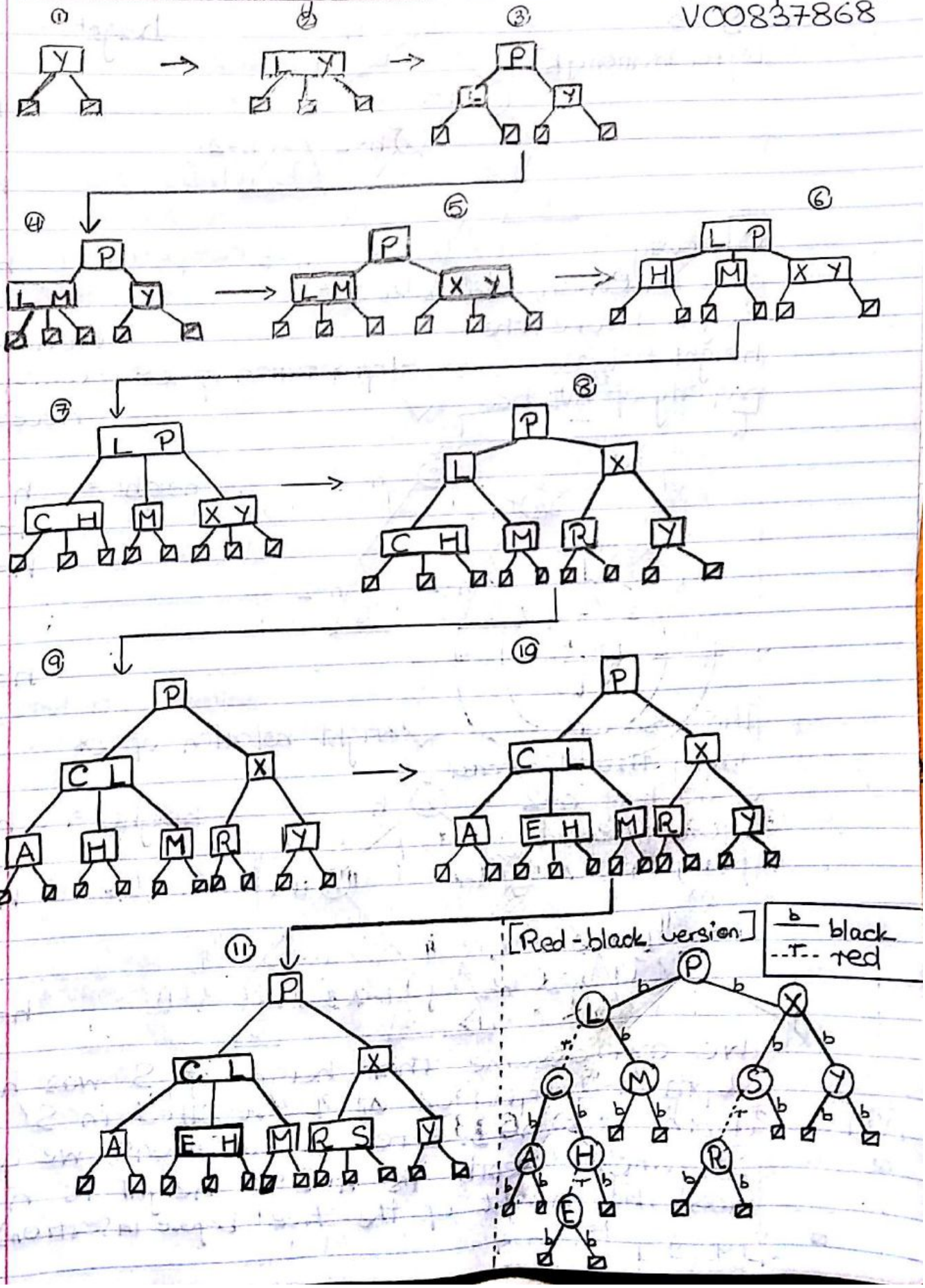
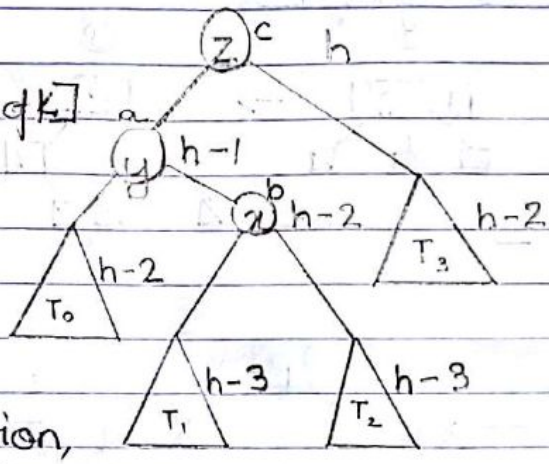
11/10/16. CSC 226 - Assignment 2

Devroop Banerjee
V00837868

1.



2. S' ⇒



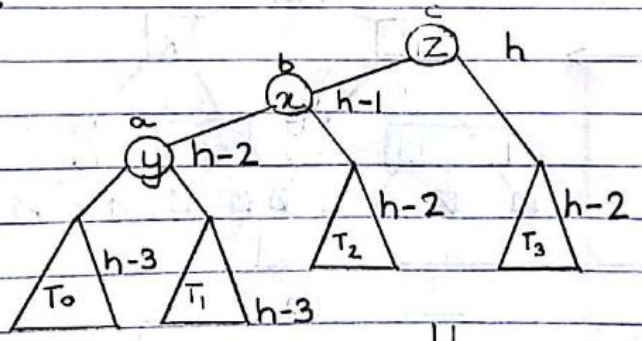
height = h
h-1
h-2
h-3

[after insertion of k]

Obviously after insertion, key k broke the height balance property of AVL tree T.

left rotation of x

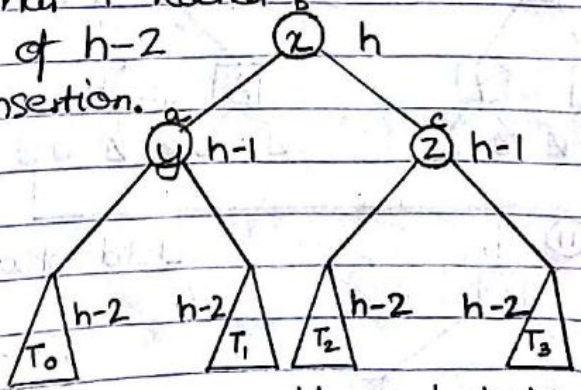
Hence restructuring was necessary.



height = h
h-1
h-2
h-3

This should imply that T reached a depth of h-2 before insertion.

right rotation of x



height = h
h-1
h-2.

We can assume that height of S was h and depth h-2. Addition of k resulted in S' whose depth was h-3. After 2 rotations, we get S*, whose depth is h-2 & height is h (which was the height of the tree before insertion).

(*) After initializing a global counter "inversions" I would add "inversions++;" if cmp is lesser than / greater than 0 and "inversions++" under "h = rotateLeft(h);" / "h = rotateRight(h);".

5. The output for the given test files were 50% for "test10.txt", 9% for "test100.txt" & 1.5% for "test1000.txt". This made it clear that the smaller the tree was and the more ordered the input values were, the lesser the percentage of red nodes. However upon using the test files from assignment 1, I realised that my results hovered around 25% of the nodes being red. These files had more random, larger, repeated values which tested my code more thoroughly. Hence I can conclude that around 25% of the nodes in a red black tree are red.

3. Considering a list of numbers is already sorted, the minimum number of inversions would be 0.
Now if we're given a list of numbers where EVERY value is out of order then every number (n-1), (n-2), ..., 2, 1 needs to be inverted. The maximum # of inversions would be $(n-1)+(n-2)+(n-3)+\ldots 2+1 \Rightarrow \frac{n\cdot(n-1)}{2}$ [as in insertion sort].

4. Red black trees have a running time of O(log n) for search, insert, and delete. To compute the # of inversions in a red black tree I would initialize a counter and increment it everytime nodes were rotated or swapped, etc. Since there are n values, and search/insert/delete take O(log n), computing the inversions would take O(n log n) time.

(*)