**Sam Lazrak**

**CS 303 Algorithms and Data Structures**

**Homework Assignment 9**

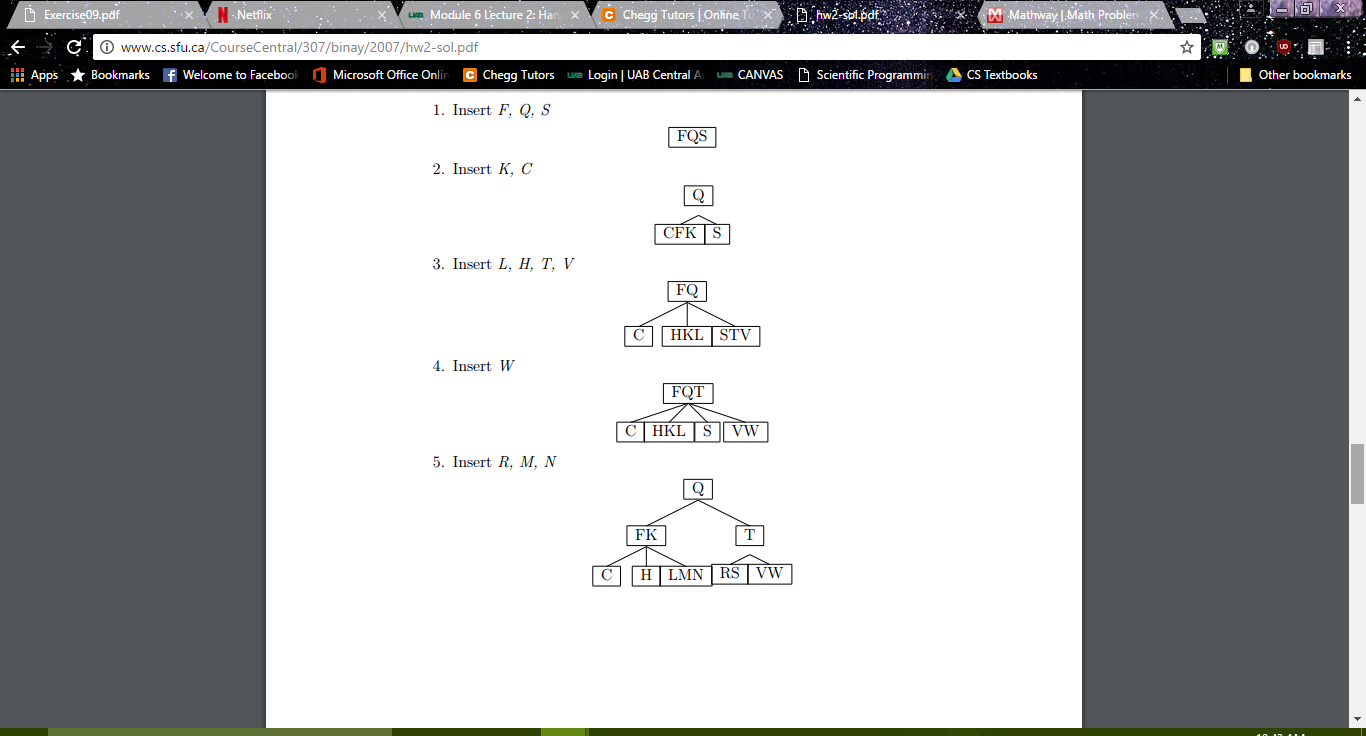
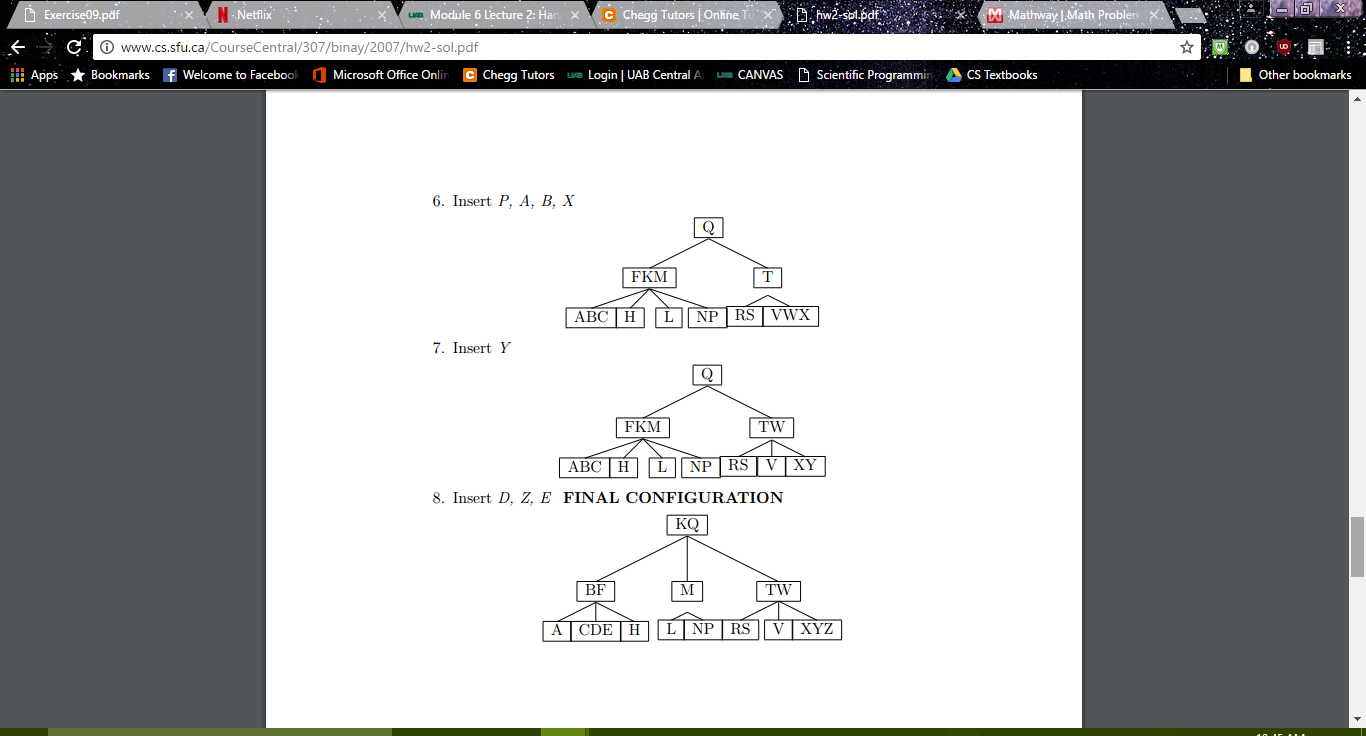
**3/27/18**

1. Work the following Exercises from Chapter 18 of the text:
   1. (3 points) Exercise 18.1-2, page 490.

* Property 5 of a B-tree states that nodes have lower and upper bounds on the number of keys they can contain. We express these bounds in terms of a ﬁxed integer t ≥ 2 called the minimum degree of the B-tree:
* Every node other than the root must have at least t - 1 keys. Every internal node other than the root thus has at least t children. If the tree is nonempty, the root must have at least one key.
* Every node may contain at most 2t - 1 keys. Therefore, an internal node may have at most 2t children. We say that a node is full if it contains exactly 2t - 1 keys.

In Figure 18.1, the number of keys of each node (except the root) is either 2 or 3. So to make it a legal B-tree, we need to guarantee that t – 1 ≤ 2 and 2 t – 1 ≥ 3, which yields 2 ≤ t ≤ 3. So, t can be 2 or 3.

* 1. (3 points) Exercise 18.1-5, page 491.
* After absorbing each red node into its black parent, each black node may contain 1, 2 (1 red child), or 3 (2 red children) keys, and all leaves of the resulting tree have the same depth, according to property 5 of red-black tree (For each node, all paths from the node to descendant leaves contain the same number of black nodes). Therefore, a red-black tree will become a B-tree with minimum degree t = 2, i.e., a 2-3-4 tree.
  1. (12 points) Solve Exercise 18.2-1, page 497.



* 1. (6 points) Exercise 18.2-3, page 497.
* Minimum Key: Start from the root, traverse down the left most child until you reach a leaf. Take the left most element as the minimum key.

Predecessor: There are a few cases we have to consider when attempting to find an element’s predecessor:

* + 1. The element we selected is in a leaf
* If the element is not the left most element in the node, return the element directly to the left.
* If the element is the left most element in the node, go up the tree until you find an element in the parent node to the left of the link and return this element as the predecessor. If you reach the root and find no such parent then you attempted to find the predecessor of the minimum element, which does not exist, so return null.
  + 1. The element we selected is in an inner node
* Traverse to the left child node of the element. If you are in a leaf, return the right-most element in the leaf, otherwise traverse down rest of the tree by the right child until you reach a leaf node and then return the right-most element.
  1. (3 points) Exercise 18.3-1, page 502.

