# Assignment 1

*(may be done by a team of at most two students)*
Assigned: Monday, September 9, 2019
Due Date (both parts): Monday, September 23, 2019 (11:59 pm)
*Online Code Submission Instructions will be Posted*

## PART 1:  Define Delete in Class AbsTree

Lecture 3 presents an object-oriented definition of a binary search tree called class AbsTree, with two subclasses, Tree  and DupTree.  In this part of the assignment, you are to complete the definition of the delete operation in class AbsTree in such a way that it works for both trees and duptrees.   The method delete(int n) should ensure that, if n is removed from a tree or duptree,  the remaining values maintain the binary search tree property.  For duptrees, the removal of n from the duptree takes place only when its associated count is 1.

Starter code for this part is given at  Resources → Assignments → Tree_Test.java.  You are to extend this code in order develop your solution.

***Preliminaries***:  In order to facilitate a simpler solution for delete, a protected field AbsTree parent is added to class AbsTree.   Revise the insert method so that the parent field is correctly set when a value is inserted into a tree or duptree.

Define three procedures, min(), max(), and find(n) which return, respectively, the AbsTree node with the minimum value, the AbsTree node with the maximum value, and the AbsTree node containing the value  n.  If n is not present in the tree, find(n) should return null.

Similar to insert, the code for delete is kept entirely in class AbsTree and captures what is common to trees and duptrees.   The differences in delete's behavior between Tree and DupTree are expressed in terms of two protected abstract methods, called get_count() and set_count(int n), with obvious meanings.  You should implement these two methods in classes Tree and DupTree.

***Defining delete***:   A good explanation of delete is given at:

http://www.algolist.net/Data_structures/Binary_search_tree/Removal

There are four main cases to delete depending upon whether the value to be deleted is at:

   (i)  A leaf node (but not the root); or
   (ii) A non-leaf node (but not the root) with only one non-null subtree; or    internal node
   (iii) A root node with one non-null subtree; or
   (iv) A node with both non-null subtrees.   internal node

Note that we do not allow the root node to be deleted if it has both null subtrees.  Also,  when a value n  in a duptree has a count > 1, the method delete(n) should decrement the count field associated with n  but should not delete the node.  If a value n  is associated with a count == 1, the method delete(n) should remove the node containing value n from the duptree.

The code for the top-level definition of `delete` in class `AbsTree` has been provided to you – do not modify this definition. The method makes use of four helper methods, called `case1`, `case2`, `case3L`, and `case3R`. The methods `case1` and `case2` correspond to cases (i) and (ii) described above while case (iii) is implemented using two methods, called `case3L` and `case3R`, depending upon whether the missing tree is on the left (`case3R`) or the right (`case3L`). Case (iv) can be handled using `case3L` or `case3R` – the starter code given to you makes use of `case3R`.

The file `Tree_Test.java` provides two tester classes: `Tree_Test`, for testing trees, and `DupTree_Test`, for testing duptrees. Run both test cases to completion, and save the diagrams in files called `A1_Tree_obj.png` and `A1_Tree_seq.png` for object and sequence diagrams for trees; and `A1_DupTree_obj.png` and `A1_DupTree_seq.png` for object and sequence diagrams for duptrees, respectively. Choose the "Objects with Tables" option for the Object Diagrams.

**What to Submit**: Prepare a top-level directory named *A1_Part1_UBITId1_UBITId2* if the assignment is done by two students (list the *UBITIds* in alphabetic order); otherwise, name it as *A1_Part1_UBITId* if the assignment is done solo. Note: your UBITId is **not** your 8-digit person number; it is your login id.

In the top-level directory, place `Tree_Test.java` and all object and sequence diagrams. Compress the directory and submit the compressed file using the Linux command `submit_cse522`. Details regarding online submission will be posted at

Resources → Assignments → Online_Submission_CSE522.pdf

Only one submission per team is required. A screen-cast, `A1_Part1.mp4`, will also be posted to clarify delete's behavior.

*Important Note:* Do not change the names of classes, fields, or methods given in the Assignment. Do not change the names or the number of parameters for the methods. You are free to have any number of local variables in the methods that you have been asked to define.


## PART 2: Transform Inheritance in terms of Delegation

Will be posted in a couple of days.


## End of Assignment 1