Assignment 3

(may be done by a team of at most two students)
Assigned: Monday, October 14
Due: Wednesday, October 30, 11:59 pm (parts 1 and 2)

Reminder: Mid-Term Exam, Tuesday, October 22 (in class)

Part 1: Contracts for trees and iterators

Posted under `Resources→Assignments` is a zip file `Tree_Contracts.zip`. Unzip this file to obtain a directory `Tree_Contracts` which can be imported into Eclipse by doing *File → Import → Existing Projects into Workspace*. This project is configured so as to enable *CoFoJa* contracts in the Java source code. To run the program, you need write the missing codes and contracts, and also add
`-javaagent:lib/cofoja+asm-1.3.1-20170424.jar`
in the *VM arguments* textbox under *Run Configurations*, as illustrated in Lecture 13, slide 25.

The project contains the familiar `AbsTree-Tree-DupTree` classes and also `AbsTree_Iterator` for iterating over trees and duptrees. Class `AbsTree` has been changed slightly (from earlier versions) by eliminating the use of abstract methods – a feature that appears to not work under *CoFoJa*.

Your task in this part of the assignment is to write *CoFoJa* contracts for the `insert` and `delete` methods of class `AbsTree` and also for the constructor, the `next()` and `stack_left_spine()` methods of class `AbsTree_Iterator`.

Note that the post-conditions for `insert(n)` and `delete(n)` need to ensure that the counts are appropriately updated for the object containing the value `n`. Hence, it is helpful to introduce in classes `Tree` and `DupTree` the methods `insert` and `delete` which delegate the actual task of insertion and deletion to their respective superclass methods, but add a small amount of code to support the enforcement of the post-conditions.

What to Do. The missing *Java* codes and *CoFoJa* contracts are indicated via comments and underlined blanks, and these are the places where you need to make changes. *Do not modify other parts of the file.* More specifically, in `AbsTree`, `Tree`, and `DupTree` you need to:

a. Define the `boolean` method `member(int n)` in class `AbsTree` by filling in the blanks.
b. Define `@Requires` for `AbsTree.delete(int n)` so that it applies to trees and duptrees.
c. Define `@Ensures` for `AbsTree.find(int n)` by stating its output condition.
d. Define `@Ensures` for `Tree.delete(int n)` by stating that `n` is no longer in the tree.
e. Define `@Ensures` for `DupTree.insert(int n)` and `DupTree.delete(int n)` suitably.

In `AbsTree_Iterator`, you need to:

a. Define `@Requires` and `@Ensures` for the constructor. The former should state that the input tree is ordered and the latter should state that the stack-top has the minimum value in the tree.

b. Define `@Requires` and `@Ensures` for `next()`. The former should state that there are more values in the tree/duptree, and the latter that the next value maintains the ascending order.

c. Define `@Ensures` for `stack_left_spine()` by stating that the next smallest value in the tree/duptree is at the top of the stack.

Run `Tree_Contracts.java` augmented with your contracts and ensure that the program works correctly. (Run using *Run Configurations*, not *Debug Configurations*.)

***What to Submit*.** Prepare a top-level directory named *A3_Part1_UBITId1_UBITId2* if the assignment is done by a team of two students; otherwise, name it as *A3_Part1_UBITId* if the assignment is done solo. (Order the *UBITId*s in alphabetic order, in the former case.) In this directory, place the revised `Tree_Contracts.java`. Compress the top-level directory and submit the compressed file using the `submit_cse522` command. Only one submission per team is required.

Part 2: JUnit Testing

Will be posted later.

End of Assignment 3 Part 1