Assignment 3

(may be done by a team of at most two students)

Due Date for Part 2: November 1, 11:59 pm
Due Date for Part 1: October 30, 11:59 pm (same as before)

Part 2: JUnit Test Suite for Tree and DupTree

Posted under `Resources→Assignments` is a zip file `JUnit.zip` containing five files: `BST.java`, `BST_Tree_Test.java`, `BST_DupTree_Test.java`, `AllTests.java` and `Sample_Output_Part2.txt`.

The file `BST.java` contain three classes to be tested: `Tree`, `DupTree`, and `TreeIterator`. The files `BST_Tree_Test.java` and `BST_DupTree_Test.java` show the overall outline of the code to be developed by you.

How to develop `BST_Tree_Test.java`**:**

- `setup()`: Build a tree by inserting *25 random numbers* in the range `0..24` into it. Also record these numbers in a Java `TreeSet` object.

- `check_invariant()`: Use *`assertTrue`* to check the binary search tree property. You need to define the boolean `ordered()` function, as illustrated in class.

- `test_insert()`: Create two iterators, one for `Tree` and the other for `TreeSet`, and check using *`assertTrue`* that every number yielded by one iterator is also yielded by the other. This ensures that `insert` has inserted all the numbers correctly (and no more).

How to develop `BST_DupTree_Test.java`**:**

- `setup()`: Build a duptree by inserting *25 random numbers* in the range `0..24`. Also record these numbers in a Java `ArrayList` object. Sort the array list after all numbers are added.

- `check_invariant()`: Use *`assertTrue`* to check the binary search tree property. You need to define the boolean `ordered()` function, as illustrated in class.

- `test_insert()`: Create two iterators, one for `DupTree` and the other for `ArrayLisy`, and check using *`assertTrue`* that every number yielded by one iterator is also yielded by the other. This ensures that `insert` has inserted all the numbers correctly (and no more).

- `test_delete()`: Insert a random number `v` into the duptree in the range `0..24`. Obtain the count associated with `v` using `get_count()` – this function to be written by you. Next, delete `v` from the duptree and check that the count has decreased by one if `v`'s original count was more than one; otherwise, check that `v` is no longer present in the duptree.

Once developed, run the project as a *JUnit Test Suite* and check that you get an output similar to what is illustrated in the file `Sample_Output_Part2.txt`. Name your output file `Output_Part2.txt`. Note: Since random numbers are to be inserted into the tree and duptree, you are likely to have different lists of numbers in your console output.

The file `AllTests.java` creates the JUnit Test Suite – no changes are needed to this file.

***What to Submit*.** Prepare a top-level directory named *A3_Part2_UBITId1_UBITId2* if the assignment is done by a team of two students; otherwise, name it as *A3_Part2_UBITId* if the assignment is done solo. (Order the *UBITId*s in alphabetic order, in the former case.) In this directory, place the five files: `BST.java`, `BST_Tree_Test.java`, `BST_DupTree_Test.java`, `AllTests.java`, and `Output_Part2.txt`. Compress the top-level directory and submit the compressed file using the `submit_cse522` command. Only one submission per team is required.

End of Assignment 3 Part 2