Homework # 9
due Monday, November 12, 10:00 PM

In the previous assignment, we implemented the ApptBook ADT using a binary search tree (BST), but left removal undone. In this assignment, we will update the data structure to have a “parent” pointer and add removal capability.

1 Concerning the Data Structure

We will update each tree node to have a “parent pointer” as explained in the Navigating Trees handout. That handout describes how to use the parent pointer to get to the “next” node. Use this to make “advance” simpler and faster.

When the tree is changed (when adding or removing nodes), you will need to update the parent pointers. You also need to update the invariant checker (wellFormed as its helper methods) to check that they are set correctly. We have updated the invariant checker tests to address the change in data structure.

2 Concerning Removal

If the node being removed has a right child, then the immediate successor (the same one found in “advance”) can be used to replace this node’s data. Otherwise (if there is no right child), the left child can replace this node entirely. Draw pictures to help you handle the replacement correctly. Our solution uses a helper method “yieldPlaceToChild” to reduce the complexity.

3 The Test Suite

We provide a full complement of tests:

TestApptBook Our ADT test suite.
TestInvariantChecker Updated data structure checker tester.
TestEfficiency Updated efficiency tests.
RandomTest The random test now includes removal. As before, it is available in the JAR.

4 What you need to do

You need to

• Add parent pointers.
• Check parent pointers.
• Update advance to use parent pointers.
• Implement removeCurrent.
• Update insert and clone to handle parent pointers.