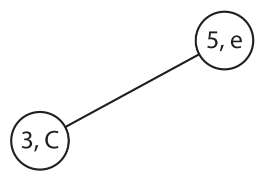
CPSC 331 — Solutions for Tutorial Exercise #9

Introduction to Binary Search Trees

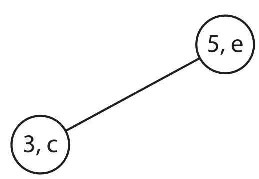
1. In this case you were asked to consider a binary search tree representing a partial function $f : \mathbb{Z} \to \texttt{Char}$, where `Char` is the set of characters that can be represented using Java's `Character` class. You were asked to perform a sequence of `set` operations beginning with the following binary search tree — which corresponds to the partial function $f : \mathbb{Z} \to \texttt{Char}$ such that $f(3) =$ "C", $f(5) =$ "e', and $f(x)$ is undefined for every integer $x \in \mathbb{Z}$ except for 3 and 5:



Note This problem should be solved by tracing the execution of the "`set`" algorithm described in Lecture #10.

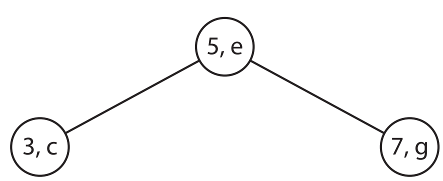
(a) `set`(3, "c")

Solution: The value at the node storing key 3 would be changed to "c":



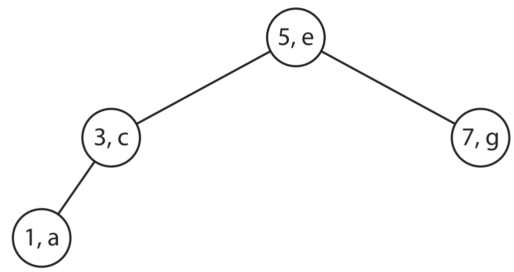
(b) `set`(7, "g")

Solution: A node storing 7 and "g" would be added as the right child of the node storing key 5:



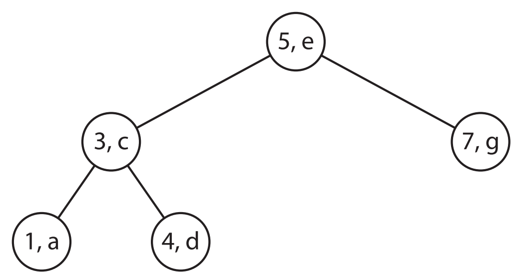
(c) $\texttt{set}(1, \text{“a”})$

Solution: A node storing 1 and “a” would be added as the left child of the node storing key 3:



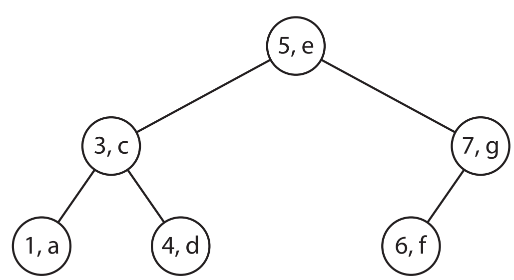
(d) $\texttt{set}(4, \text{“d”})$

Solution: A node storing 4 and “d” would be added as the right child of the node storing key 3:



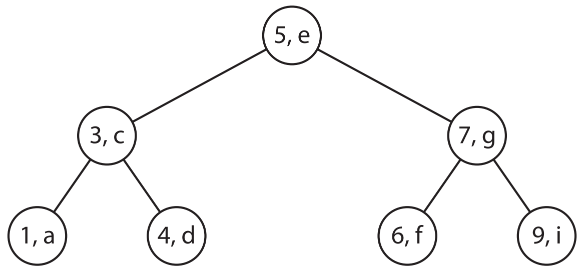
(e) $\texttt{set}(6, \text{“f”})$

Solution: A node storing 6 and “f” would be added as the left child of the node storing key 7:



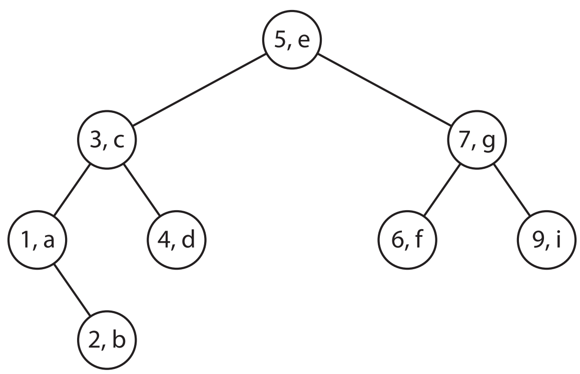
(f) `set`(9, “i”)

Solution: A node storing 9 and “i” would be added as the right child of the node storing key 7:



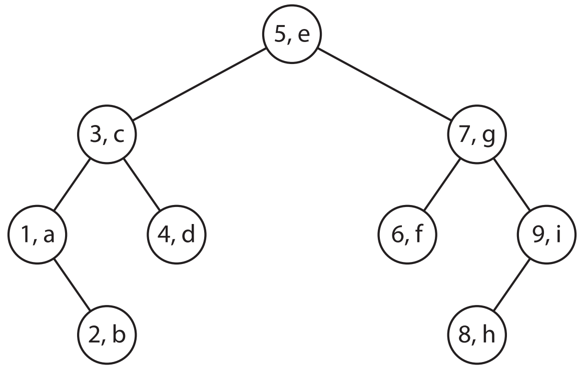
(g) `set`(2. “b”)

Solution: A node storing 2 and “b” would be added as the right child of the node storing key 1:



(h) `set`(8, “h”)

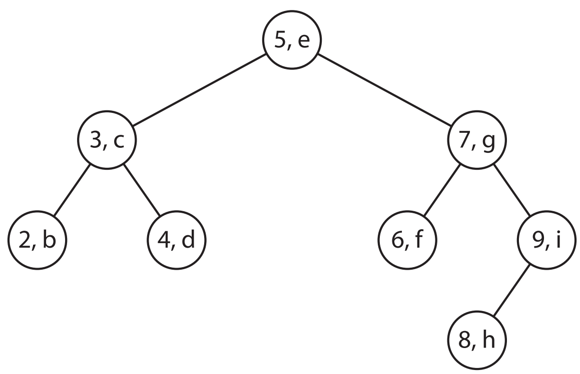
Solution: A node storing 8 and “h” would be added as the left child of the node storing key 9 — resulting in the binary search tree shown at the end of this question on the tutorial exercise:



2. You were next asked to continue with the above binary search tree the binary search trees that would be obtained by executing the following sequence of “`remove`” and “`set`” operations.

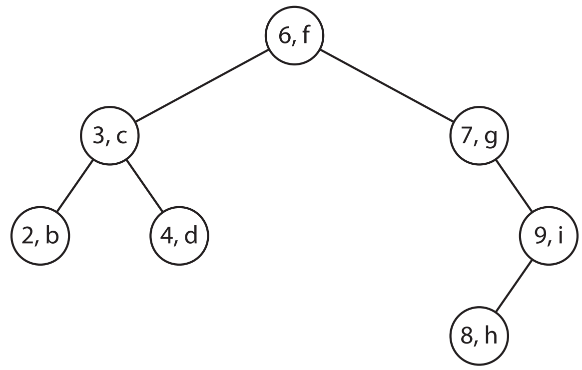
    (a) `remove`(1)

    Solution: Since the node storing key 1 has no left child (that is, this child is `null`) and a (non-`null`) right child, the right child is promoted:

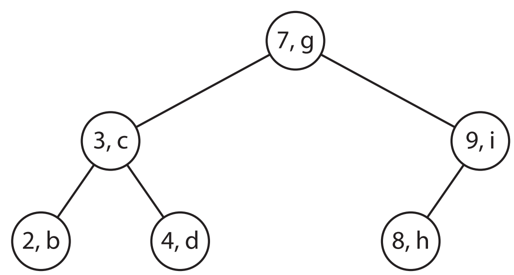
    (b) `remove`(5)

    Solution: Since the node storing key 5 has two non-`null` children, the contents of its “successor” — the key 6 and value “f” — should be copied into it The node originally storing these values is a leaf so that it can simply be deleted:



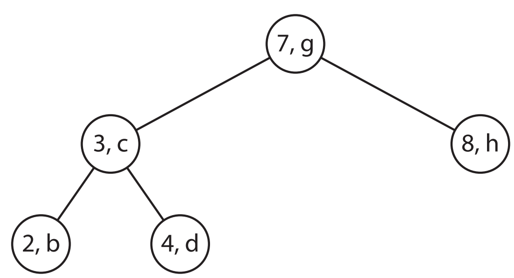
(c) `remove`(6)

Solution: Since the node storing key 6 has two non-`null` children, the contents of its "successor" — the key 7 and value "g" — should be copied into it. The node originally storing these values has a non-`null` right child but not a (non-`null`) left child, so that its right child should be promoted in order to delete it:



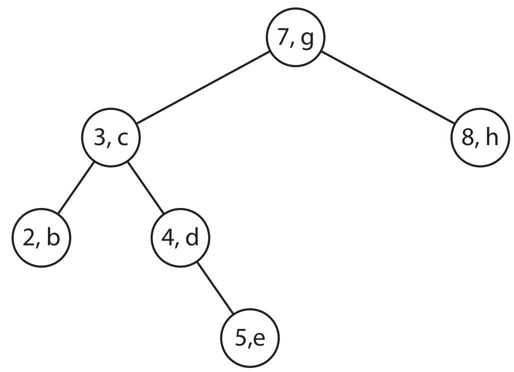
(d) `remove`(9)

Solution: Since the node storing key 9 has a non-`null` left child but no such right child, its left child should be promoted in order to delete it:



(e) $\texttt{set}(5,$ “e”$)$

Solution: A node storing key 5 and value “e” should be added as the right child of the node storing key 4:



3. You were asked to modify the program `BSTDictionary.java` in order to produce a program `BSTOrderedSet.java` providing a class that implements an “Ordered Set” using a binary search tree instead of `Dictionary`.

   Solution: A program `BSTOrderedSet.java` is now available.

   While the names of methods to be provided and the outputs to be returned (notably including when exceptions should be thrown) are different — and pairs of “keys” and “values” should be replaced by individual “elements” — the overall logic and structure of corresponding algorithms does not need to significantly change.