CIS2168 006 Assignment 6
Integer Collection Using Binary Search Tree

1. Objectives

This assignment will help you to:

- Learn how to program using data structure Binary Search Tree
- Understand how Binary Search Tree works
- Enhance your skills in programming using linked lists

2. Overview

You write a Java application simulating collections of simple integers in **`int`** type.

You will use Binary Search Trees to store the **`int`** values. The manner in which these values are inserted, deleted, and accessed is the same as a Binary Search Tree (BST) data structure. You must revise the class BSTIntCollection I gave you as required such that it is a fully functional Binary Search Tree for only `int` values and has the implementation of ALL required methods. **You CANNOT change the definition of the node class BSTIntNode.**

You must also write another class BSTIntCollectionTest that uses the class BSTIntCollection and completes a number of operations on a single BST integer collection and multiple BST integer collections.

More specifically your BSTIntCollection class must contain the following methods:

- Add: add a given integer into a BST integer collection.
- Contain: check if a BST integer collection contains a given target.
- Delete: delete a given integer from a BST integer collection. You can use the inorder predecessor to replace the value in the deleted node.
- (Bonus) Delete2: delete a given integer from a BST integer collection. Must use the leftmost node data in the right subtree of the deleted node. That is use the inorder-traversal successor of the deleted node to replace the integer in the deleted node.
- toString(): return a string that contains the preorder traversal of a BST integer collection. Must have each node on a single line and indent the local root 2 spaces further at each increasing level. The return string shows the structure of the entire tree.
- inOrderTraversal(): returns the inorder traversal string of a BST integer collection. Must separate each two adjacent values by 2 spaces.
- (Bonus) postOrderTraversal():returns the postorder traversal string of a BST integer collection. Must separate each two adjacent values by 2 spaces.
- getHowMany(): return the number of nodes in a BST integer collection.
- copy: copy the integers in a given BST integer collection to the calling BST integer collection

- equals: check if a given BST integer collection is exactly the same as the calling BST integer collection.

More specifically your BSTIntCollectionTest will create three BST integer collections. Use the first collection to test the implementation of all methods except copy and equals. Then use the first and the second collections to test equals. Then use the second and the third to test copy.

3. Implementation Requirements

- When you revise BSTIntCollection class, you **cannot change the definition** of the nested class BSTIntNode and the header of class BSTIntCollection.
  - This requirement means that you **cannot use Java API** class for Binary Search Tree.
- You must implement the following methods as stated in the BSTIntCollection class.
  - public boolean add(int intData)
    i. Add the given integer data into the calling BSTIntCollection object. Return true if success, otherwise false.
  - public boolean contain(int intTarget)
    i. Try to find the target in the calling BSTIntCollection object. Returns true if it's found. Otherwise, return false.
  - public boolean delete(int intTarget)
    i. Delete the given integer from the calling BSTIntCollection object. You can use the inorder predecessor to replace the value in the deleted node.
  - (Bonus) public boolean delete2(int intTarget)
    i. Delete the given integer from the calling BSTIntCollection object. You **MUST use the inorder successor to replace the deleted node's integer value**. The inorder successor of a node is the leftmost element in the right subtree of this node.
  - public String toString()
    i. Very similar to toString() in BinaryTree class covered in the lecture section
    ii. Returns a string that contains the preorder traversal of the calling BSTIntCollection object. The traversal output indent each local root at a distance proportional to its depth. Indent 2 spaces at each increasing level. If a subtree is empty, display null. See Page 311 for a sample format.
  - public String inOrderTraversal()
    i. A method that returns the inorder traversal string of the calling BSTIntCollection object. The string is a sequence of integer values, separated by two spaces between each two adjacent values.
  - (Bonus) public String postOrderTraversal()
    i. A method that returns the postorder traversal string of the calling BSTIntCollection object. The string is a sequence of integer values, separated by two spaces between each two adjacent values.
  - public int getHowMany()

        i. Return the number of nodes in a BST integer collection
    - public void copy(BSTIntCollection bstIntCollectionP)
        i. Copy the integers in the given BSTIntCollection object referenced by bstIntCollectionP to the calling BSTIntCollection object.
        ii. Add the integers in the bstIntCollectionP in their **preorder traversal** sequence to the calling BSTIntCollection object
    - public boolean equals(BSTIntCollection bstIntCollectionP)
        i. Check the given BSTIntCollection object referenced by bstIntCollectionP is exactly the same as the calling BSTIntCollection object. That is they both the exact same structure. You MUST implement everthing from scratch. You CANNOT call other methods to implement this method.
- The class BSTIntCollectionTest must present to the user a text-based menu, which includes these operations
    - Create an empty BSTIntCollection
    - Insert an **`int`** value to the BST integer collection
    - Remove a given **`int`** value from the BST integer collection
    - (Bonus) Remove2 a given **`int`** value from the BST integer collection
    - Check if a given **`int`** value is contained in a BST integer collection
    - Get the total number of **`int`** values in a BST integer collection
    - Display all elements in the BST integer collection, showing the structure of the tree (result of calling toString())
    - Inorder traverse a BST integer collection
    - (Bonus) Postorder traverse a BST integer collection
    - Check if two BST integer collections are exactly the same
    - Copy a BST integer collection to another BST integer collection
    - Quit from the program
- Your class BSTIntCollectionTest must create 3 BST integer collections.
- **You earn bonus points only if you get all required methods correctly implemented.**

4. Major Steps

- Understand the related classes I gave you in previous lectures (Lec#8, Lec#9).
    i. BinaryTree1.java, BinarySearchTree.java, SearchTree.java, ReadBinaryTree1.java
- Revise the class BSTIntCollection. Add necessary data fields and methods. At one method at a time.
- Write the class BSTIntCollectionTest.
    i. First use fixed values
    ii. Then add the menu

5. Detailed Hints

- You need to add at least one constructor to the BSTIntCollection class
- Add, delete, contain are similar to the methods add, delete, find in BinarySearchTree class.

- toString is very similar to the method toString in BinaryTree class.

6. Submission Requirements & Grading

This assignment is **due by 11:50PM, Thursday, October 29, 2015**.