CSCI 241, Fall 2012

Assignment #2: Balanced Binary Search Trees (30 pts)

Summary

- Goal: This assignment is worth 12% of the grade for the course. In this assignment you will implement balanced search tree data structures. Please show some independence in completing this assignment, and continue to follow good design practices!
- Due time: **end of day (11:59pm), Monday, Oct 29.**
- Hand in: submit your file(s) via the **Assignment 2 Submission** link on the Blackboard under "Assignments". A Readme file is not required but you may include one if there's anything you want to tell me.

Task

Starting with an empty balanced binary search tree, insert the following keys into the tree in the given order

```
Keys: 2   3   5   4   7   8   9   1
```

And then delete the keys 5 7 in the given order from the tree.
On deleting an internal node, a substitution will be chosen from the RIGHT (inorder successor), if needed.

Assume that the balanced tree is implemented as one of the following:

a) AVL tree
b) Red-black tree

Pick one to implement and show the underlying tree after each insertion and deletion.

Input and Output (15 pts)

Write an Ada programs for accepting insertion, deletion, and print requests on an implemented balanced tree of positive integers from standard input.

Formats for a red-black tree (no color field output for an AVL tree):

```
Please enter your request (i=insert, d=delete, p=print, e=exit): i 20

Please enter your request (i=insert, d=delete, p=print, e=exit): p
black(null,20,null)

Please enter your request (i=insert, d=delete, p=print, e=exit): i 10

Please enter your request (i=insert, d=delete, p=print, e=exit): p
black(red(null,10,null),20,null)
```

```
Please enter your request (i=insert, d=delete, p=print, e=exit): i 5

Please enter your request (i=insert, d=delete, p=print, e=exit): p
black(red(null,5,null),10,red(null,20,null))

Please enter your request (i=insert, d=delete, p=print, e=exit): i 25

Please enter your request (i=insert, d=delete, p=print, e=exit): p
black(black(null,5,null),10,black(null,20,red(null,25,null)))

Please enter your request (i=insert, d=delete, p=print, e=exit): i 15

Please enter your request (i=insert, d=delete, p=print, e=exit): p
black(black(null,5,null),10,black(red(null,15,null),20,red(null,25,nul
l)))

Please enter your request (i=insert, d=delete, p=print, e=exit): d 5

Please enter your request (i=insert, d=delete, p=print, e=exit): p
black(black(null,10,null),15,black(null,20,red(null,25,null)))

Please enter your request (i=insert, d=delete, p=print, e=exit): e
Bye!
```

Implementations (15 pts):

1. Write a generic AVL tree package if you choose implementing using AVL trees or a generic red-black tree package if you use red-black trees.
2. Resources you can use and modify:
   - Resources indicated on the syllabus**.**
   - Sample codes on http://users.cis.fiu.edu/~weiss/ada.html
3. Comment on your submission properly.
4. Run the following two test cases separately on your implementation, from an empty tree,
   - `Insert 5, 16 in this order, delete 5.`
   - `Insert 53, 27, 75, 25, 70, 41, 38, 59, 39 in this order, delete 25, 70 in this order.`

Extra Credit – package inheritance in Ada (10 pts)

1. Start by writing a binary search tree package using the linked representation. The package should implement the basic binary search tree procedures (such as find, find_min, find_max, successor, predecessor, insert, delete, basic rotation—left_rotate, right_rotate, etc.).
2. Write the AVL tree package or the red-black tree package as a child package of the binary search tree package.
   Reference: http://www.radford.edu/~nokie/classes/320/child.packages.html

3. If this inheritance version is your final implementation, submit this version only. That is, you do not need to implement both versions -- the non-inheritance one and the inheritance one.