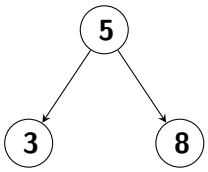
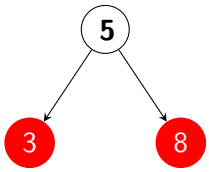
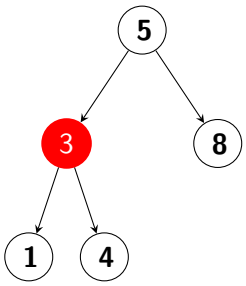
CSC 330 Advanced Data Structures
Fall 2014 - Project
Check-In I: Friday, October 10, 2014 8:00am
Check-In II: Friday, October 24, 2014 8:00am
Check-In III: Friday, October 31, 2014 8:00am
Final Submission Due Friday, November 14, 2014, 8:00am

Project Background

A Red-Black Tree is a binary search tree where, every node is colored either Red or Black. The root node is always Black. If a node is Red, it's children must be Black, that is no Red node may have a Red child. Additionally, every path from a node to a leaf must contain the same number of black nodes. The following are examples of Red-Black Trees.







Project Requirements

You must name all of the classes and methods as shown below. Failure to match these names will result in failing portions of the tests, ultimately impacting your grade for this project.

Create an Element class named `Element`. An element consists of a (key, value) pair. Provide any methods necessary.

Create a Red-Black Node class named `RedBlackNode`. You Red-Black Node class should have the following member variables and methods:

```
public class RedBlackNode {
    private Element element;
    private RedBlackNode left;
    private RedBlackNode right;
    private int color;

    public RedBlackNode(Element e, RedBlackNode l, RedBlackNode r) {
                        element = e;
                        left = l;
                        right = r;
                }
    public Element getElement() { /*To-Do*/ }
    public int getColor() { /*To-Do*/ }
    public RedBlackNode getLeftChild() { /*To-Do*/ }
    public RedBlackNode getRightChild() { /*To-Do*/ }
}
```

Create a Red-Black Tree class named `RedBlackTree`. Your Red-Black Tree Class must have a public default constructor. Your Red-Black Tree class will need to implement the following interface:

```
public interface TreeInterface {

    public String find(int key);

    public void insert(int key, String value);

    public void delete(int key);

    public String remove(int key);
}
```

The find method should search the Red-Black tree and return the value associated with the given key. If the key is not in the tree, return null.

The insert method should insert the given (key, value) pair in the Red-Black Tree via top-down insertion. If the key already exists, the existing value should be overwritten with the new value provided.

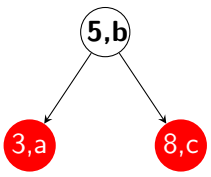
The delete method should delete the (key, value) pair with the given key from the tree via top-down deletion. If the key is not in the tree, do nothing.

The remove method like delete, removes the (key, value) pair with the given key from the tree via top-down deletion, but returns the value associated with the key. If the key is not in the tree, return null.

Your Red-Black tree class should also override the toString method to print the tree in pre-order traversal as described below. If the tree is empty the toString method should return a default string that is exactly: `This tree is empty.`

```
Start at root
prefix = "root"
while(not done)
        create line: prefix+":C(k,v)" with each (key, value) pair,
                and C is either RED or BLACK.
        visit each child in order (left-to-right)
                for the left child append "-left" to the prefix
                for the right-child append "-right" to the prefix
```

For example the tree shown would produce this output:




```
root:BLACK(5,b)
root-left:RED(3,a)
root-right:RED(8,c)
```

Red-Black Tree Validator

Given a string representation of a colored tree, we want to determine if it satisfies the properties of a Red-Black Tree.

Create a class `RBTValidator` it must contain a method called `validate` that takes a single parameter (String) and returns a boolean. If the String provided represents a tree that satisfies the properties of a Red-Black Tree return true, otherwise return false. Your RBTValidator class must have a public default constructor.

The string will be in the following format:
prefix:$C(k,v)$ **where**: prefix is the path to the node (eg `root-left-right`), C is the color of the node (either RED or BLACK), k is the key and v is the value of the node. Each node is separated by the newline character: `\n`

You may assume the string is formatted correctly, and always in Pre-Order form.

You may add additional methods and classes as you see fit.

Project Report

Produce a report with the following sections:

1. Introduction and Description of the problem
2. Discuss the properties of the Red-Black Tree data structure
3. Discuss the find operation, show that it is in O(log n).
4. Discuss the rotation operation, show that the Red-Black Tree properties are maintained after all possible rotations
5. Diagram an example Red-Black tree operation using the first 10 letters of your name as the input. Specifically, on the 1st occurence of a letter, insert that letter into the tree. On the Second occurence of a letter delete that letter from the tree, alternating insert and delete for every subsequent occurence. Be sure to diagram each intermediate change to the tree. Evaluate characters in alphabetical order, ignoring case: $a < b < c < .... < x < y < z$
6. Discuss your RBTValidator, how did you implement it? What properties did you check for? How do you ensure those properties are satisfied?
7. Personal Reflection. What parts of the assignment were difficult, which were easy? What were your initial assumptions? Were they correct? Etc.

Project Submission - Check-In Submission Submit the following files for your project-to-date on Canvas containing:

1. your source code (all files)
2. a readme.txt file
3. a pdf of your project report

readme.txt This file should detail any changes you made to the specification and your justification for making them. List any features that are missing or you have added. Provide any other comments or information you think I (the grader) or someone using your code should know.

Project Submission - Final Submission Submit the following files for your project on Canvas containing:

1. your source code (all files)
2. a readme.txt file
3. a pdf of your project report

readme.txt This file should detail any changes you made to the specification and your justification for making them. List any features that are missing or you have added. Provide any other comments or information you think I (the grader) or someone using your code should know.