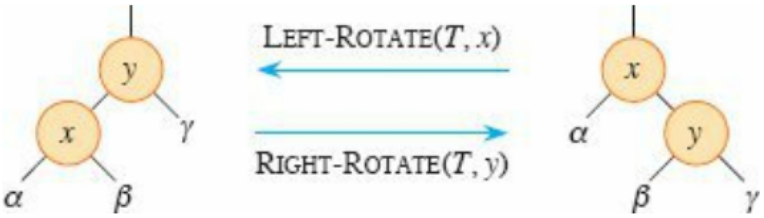
CS146: Quiz 5
Due Tuesday, February 28, at 7:00AM
10 points

You will implement five methods for Red-Black Trees. The two rotation methods, the insert and fix up insert methods, and a toString method that returns a string representation of the tree based on an in order traversal of the tree. **To get started, import the starter file, RBTree.java into the redblack package you create in a new Java Project.** Please do not change any of the method signatures in the class. Implement the methods described below. You are free to test your code however you prefer. These methods should be completed individually using any IDE you are comfortable with. You are free to use the textbook, slides, class notes, and the Java API Documentation, but **DO NOT** consult any other resources.



`void rotateLeft(Node node)`
The method completes a left rotation around Node node. (see image above). The pseudocode is available in your textbook.

`void rotateRight(Node node)`
This method completes a right rotation around Node node. (see image above). You need to figure out how to use the pseudocode from the rotateLeft method to create the rotateRight method.

`public void insert(int i)`
This method should insert a new Node into the tree containing data equal to i. **The tree DOES NOT accept duplicate values so if you try to insert a value that already exists in the tree, the tree should not change. This will require you to modify the pseudocode below in order to correctly implement the method.**

```
rbInsert(T,z)
1    x=T.root
2    y=T.nil
3    while x!=T.nil
4      y=x
5      if z.key<x.key
6      x=x.left
7      else
8      x=x.right
9    z.parent = y
10   if y == T.nil
11     T.root = z
12   else if z.key < y.key
13     y.left = z
14   else
15     y.right = z
16    z.left = z.right = T.nil
17   z.color = RED
18   rbInsertFixup(T,z)
```

`void insertFixUp(Node node)`

This method recolors and rotates the tree to ensure that the tree remains a valid red-black tree after inserting a new value. See the pseudocode below.

```
rbInsertFixup(T,z)
1 while(z.p.color == RED)
2   if z.p == z.p.p.left
3     y = z.p.p.right
4     if y.coor==REF
5       z.p.color = BLACK
6       y.color = BLACK
7       z.p.p.color =RED
8       z = z.p.p
9     else
10      if z == z.p.right
11        z = z.p
12        leftRotate(T,z)
13      z.p.color = BLACK
14      z.p.p.color == RED
15      rightRotate(T,z.p.p)
16   else
17     //z.p == z.p.p.right, swap left and right from if statement
18  T.root.color = BLACK
```

`public String toString()`

This method should return a string representation of the tree from an in order traversal. An in order traversal visits the left subtree, then the current node, then the right subtree resulting in the data being printed in sorted order. This method has to be completed iteratively (HINT: you might want to use a Stack).

Examples

- An empty tree should return the String "{}"
- A tree with the nodes inserted in the order 2, 1, 3 should return the String "{(1,RED),(2,BLACK),(3,RED)}"

Submission

Please create a jar or zip file of your project. It should include the java file (RBTree.java) and submit it on Canvas. If your jar file includes class files or other extraneous files outside of the package folder and java file, you will receive an automatic one point deduction.