ITCS 2214 Data Structures
Assignment 3

Download the assignment from Moodle. Change the name of the folder to your first-name underscore last-name underscore assign1 (FirstName_LastName_Assign3). DO NOT change the structure of the files. If you write your code in some where else, copy and paste it in the RedBlackTree.java and RedBlackNode.java files. ENSURE you DO NOT paste over the package declaration at the top of the files. DO NOT change the interface. Use Main.java as your driver; put test code in this file to debug your data structure. It is ok if you leave the code in Main.java, I will use my own code. In the Read Me file, write a description of each method you implemented (what the method is suppose to do). Upload a zip folder to Moodle before the deadline.

Fill in the methods in the RedBlackTree.java file. You may use other private helper functions as you see fit. However, NO other instance or static variables should be used. You will NOT be graded on the efficiency of your code. In other words, just get the job done. My test code will invoke operations on your RedBlackTree, checking its state.

These are the methods you should implement:

a. RedBlackTree(): creates an empty red/black tree. (1 point)

b. add(T element): add the element to the tree. (3 points)

c. levelOrderTraversal(): prints the content of the nodes in a levelorder traversal of the tree. (5 points)

d. colorTraversal(): prints the content of the nodes, and their color in parenthesis, in a traversal of the tree. Any traversal can be used. (2 points)

e. positionTraversal(): prints the content of the nodes, and their position in parenthesis, in a traversal of the tree. Any traversal can be used. (2 points)

f. depthTraversal(): prints the content of the nodes, and their color in parenthesis, in a traversal of the tree. Any traversal can be used. (2 points)

g. reverseLevelOrderTraversal(): prints the content of the nodes in a levelorder traversal starting at the farthest level and moving upward towards the root. (6 points)

h. rebalance(): rebalance/recolor the tree after an element is added. (6 points)

i. rightRotation(node): performs a right rotation around the node. (6 points)

j. leftRotation(node): performs a left rotation around the node. (6 points)

k. RedBlackNode(): creates an empty node. (1 point)

l. RedBlackNode(T element): creates a node with the specified element. (1 point)

Every node in the tree will have its color, its depth, and its position (relative too its parent). These properties must be accurately maintained for each node at all times. (8 points)