Red Black Binary Search Trees

1. Use the code accompanying this assignment: RedBlackBST.java and RED_BLACK_BST_Tester.java. I found RedBlackBST.java on the Internet. This will save you a great deal of typing. But, like all code that you get from others, there could be a couple of bugs so test it carefully and fix it if necessary. All four of the methods that you write should be at the bottom of RedBlackBST.java
2. I wrote the code for RED_BLACK_BST_Tester.java and I tested it carefully. So, I hope it doesn't have any bugs. Please avoid modifying it.
3. Write a method that counts the number of leaf nodes in the tree: *leafCount*()
4. Write a method that counts the number of three nodes in the tree: *threeNodeCount*()
5. Write a method that counts the number of leaf nodes that are three nodes: *threeNodeLeafCount*()
6. Write a method called *toStringKeysByLevel* () that returns a string representation of the keys in the tree and helps you see the red black structure of the tree. For the three-nodes, it will put an underscore next to the key depending on whether it is the smaller or larger key in the three node. This is a sample output of a red black tree in which the following keys were added: B, C, A, X.

```
B
A  _X
null null C_ null
```

   This is what the tree might look like if it were drawn spaced nicely. The dash-dash symbols - - correspond to empty spots in the tree. Your output does not have to look like the image below.

```
          B
     A         _X
  --    --   C_   --
```

7. Use the tester that I have written for you. Execute "6. Test Me" and submit the output of this test with your java code. All of the code that you write in the BST.java file should be at the bottom of the file.
8. Upload the java code and the output and no other files. Please upload the files separately instead of putting them in a zip or compressed folder.