Question 1. Range in BST

In this question you will extend the TreeMap class of the net.datastructures package provided by the authors of your textbook (Goodrich, Tamassia & Goldwasser). For your convenience, I have extracted the subset of .java files you will need, which you can download here.

Your task is to extend theTreeMap class of the net.datastructures package to allow all entries within a range of key values [k1...k2] to be found and returned as an iterable list. It is trivial to do this in O(n) time using the entries() method of the BinarySearchTree class, which returns an iterable collection of all the entries in the tree. However, you would like to do this in O(h + m) time, where h is the height of the tree and m is the number of entries in the tree with keys in the specified range.

To accomplish this, you are provided with an interface for a new BSTRange class that extends the TreeMap class. BSTRange provides 4 new methods, which you will write:

3 protected methods:

- findLowestCommonAncestor, which returns the lowest position in a subtree that is a common ancestor to all positions with keys between k1 and k2.
- findAllAbove, which finds all entries with keys greater than or equal to a specified key k, in increasing order
- findAllBelow, which finds all entries with keys less than or equal to a specified key k, in increasing order

1 public method:

- findAllInRange, which returns all entries with keys between k1 and k2, in increasing order

findAllInRange will use the 3 protected methods to efficiently accomplish the task.

In addition to the TreeMap Class, your BSTRange class will use the Entry, Position, PositionalList, and LinkedPositionalList classes provided.

Here is a test program testFindAllInRange that provides a test case. You should, however, test your program using a broader range of test cases. Pay particular attention to boundary conditions.