Lab 2. This is how I solved lab 2. You don’t have to do it this way, but this may help you if your program is not complete and you don’t know how to proceed.

Classes: I have the following classes: BTreeTest, BTree, BTreeNode, BTreeLeafNode, BTreeInternalNode.

BTreeNode is an abstract class, its members are what’s needed in both internal nodes and leaf nodes. I member referencing the BTree it’s part of, to be able to modify the tree height and/or root members. Other than the constructor, all methods do nothing and will be overridden by the subclasses.

BTreeTest, used for testing your program.

BTree. Members are M, L, root, height, firstLeaf. The first leaf reference is for when we want to go through all the leaves in order. The methods are essentially calling the methods in BTreeNode.

BTreeLeafNode. Members are L and nextLeaf. To insert, go through the sorted array of keys to find where it will be inserted. If the key is already there do nothing. Otherwise, add the key to the array. If the number of keys is larger than L, split the leaf. To split the leaf, if the parent is null, we need to create an internal node as parent and update the BTree root and height. Then, create a new leaf, copy half of the array of keys to the new leaf, update the parent (call the parent’s update method) by adding the new appropriate key and references to the two newly split leaves.

BTreeInternalNode. Members are M and the array of children. When an insertion occurs, I call a findLeaf method to find the leaf where the insertion will occur and call the insert method on that leaf. I also wrote a method to update the node with a new key, and a split method for when adding the new key makes the node to large. The update method is similar to that of adding a key to a leaf, but in addition to inserting a key in the correct position, we have to insert the references to children in the correct positions as well. The split is similar to that of splitting a leaf, but in addition to copying half of the keys to a new internal node, we have to copy the references to children as well.

Suggestion for incremental development. First write the insert method for BTreeLeafNode. Test that it works inserting up to capacity. Then write the code for splitting the leaf, creating an internal node and a new leaf. To make it work, you need to write the update method for internal nodes. Make sure it works (a root with one key and the two leaves as children.) Then make sure it works by calling printNodes. Also check that the parent references are right. As you add more keys, the root will get more children, and eventually will exceed its capacity. That’s when you need to write the code to split an internal node. Once you finish this and it works, you’re essentially finished with the main part of the program. Writing the printLeaves method should not be too hard.