Testing and Github

B-Tree-Insert and B-Tree-Insert-Nonfull from the textbook:

```
B-TREE-INSERT(T, k)
  1. r = T.root
  2. if r.n == 2t-1
  3.       s = ALLOCATE-NODE()
  4.       T.root = s
  5.       s.leaf = false
  6.       s.n = 0
  7.       s.c_1 = r
  8.       B-TREE-SPLIT-CHILD(s, 1)
  9.       B-TREE-INSERT-NONFULL(s, k)
 10. else B-TREE-INSERT-NONFULL(r, k)

B-TREE-INSERT-NONFULL(x, k)
  1. I = x.n
  2. if x.leaf
  3.       while i >= 1 and k < x.key_i
  4.             x.key_{i+1} = x.key_i
  5.             i = i – 1
  6.       x.key_i = k
  7.       x.n = x.n + 1
  8.       DISK-WRITE(x)
  9. else while i >= 1 and k < x.key_i
 10.             i = i – 1
 11.       i = i + 1
 12. DISK-READ(x.c_i)
 13. if x.c_i.n == 2t – 1
 14.             B-TREE-SPLIT-CHILD(x, I)
 15.       if k > x.key_i
 16.                   i = i + 1
 17. B-TREE-INSERT-NONFULL(x.c_i, k)
```

Building Your J-Unit Test Suite:

You can use unit testing to do test-driven development of your B-Tree class as opposed to strictly testing it with the driver. This will speed up development of the project as it enables incremental production of the B-Tree insert method, lets you debug with simpler data sets, and gives some assurance that you have a working B-Tree class for when you write the driver. A good test suite covering B-Tree insert should consider simple cases such as just filling the root, and cases that introduce complexity such as splitting. Here's a suite you should consider making:

Test 1:

Recall that a B-Tree Node can hold up to 2t-1 keys where t is the degree of the B-Tree. Plan a J-Unit test that completely fills the root of a new B-Tree. You will need to consider following:

- What degree of B-Tree you will be testing? What would be simplest?
- What keys will you add to the B-Tree?
- What parts of B-Tree-Insert and B-TREE-INSERT-NONFULL will you have to implement to pass this test?
- How many nodes will the B-Tree have after inserting the keys you plan?
- Is the order of the keys correct after inserting?

Test 2:

If you're passing your J-Unit test for test 1, your B-Tree code should be able to create a B-Tree with a single node with a full root. Consider the following for your next test:

- If you add one more key, your B-Tree should split the root. What parts of B-Tree-Insert and B-Tree-Insert-Nonfull are required to achieve this?
- How do you access a node by its index in the tree? How should you move about the B-Tree to reach this index?
- How can you store B-Tree-Nodes on disk as opposed to main memory? Research Java's ByteBuffer and RandomAccessFile classes and think about what happens when when ALLOCATE-NODE(), DISK-READ(), and DISK-WRITE() are called.

Test 3:

Initialize a degree two B-Tree and insert the keys 1, 2, 3, 4, 5 and consider:

- How many nodes will you have?
- What will the contents of each node be?

Add the key 6 to this B-Tree and consider:

- What makes the split that will occur different from the split that occurred in test 2?
- What parts of B-TREE-INSERT and B-TREE-INSERT-NONFULL must be implemented?

Test 4:

Take the resulting B-Tree of the previous test and add the keys 7 and 8. Consider the following:

- What nodes in the B-Tree are full?.
- How many levels does the B-Tree have?

Add the key 9 to this B-Tree and consider:

- Will a split occur and if so, how does it differ then the split in test 2?
- Have we considered all regions at which split could occur?

Test 5:

Take the B-Tree from the previous test and add the keys 10, 11, and 12. Consider:

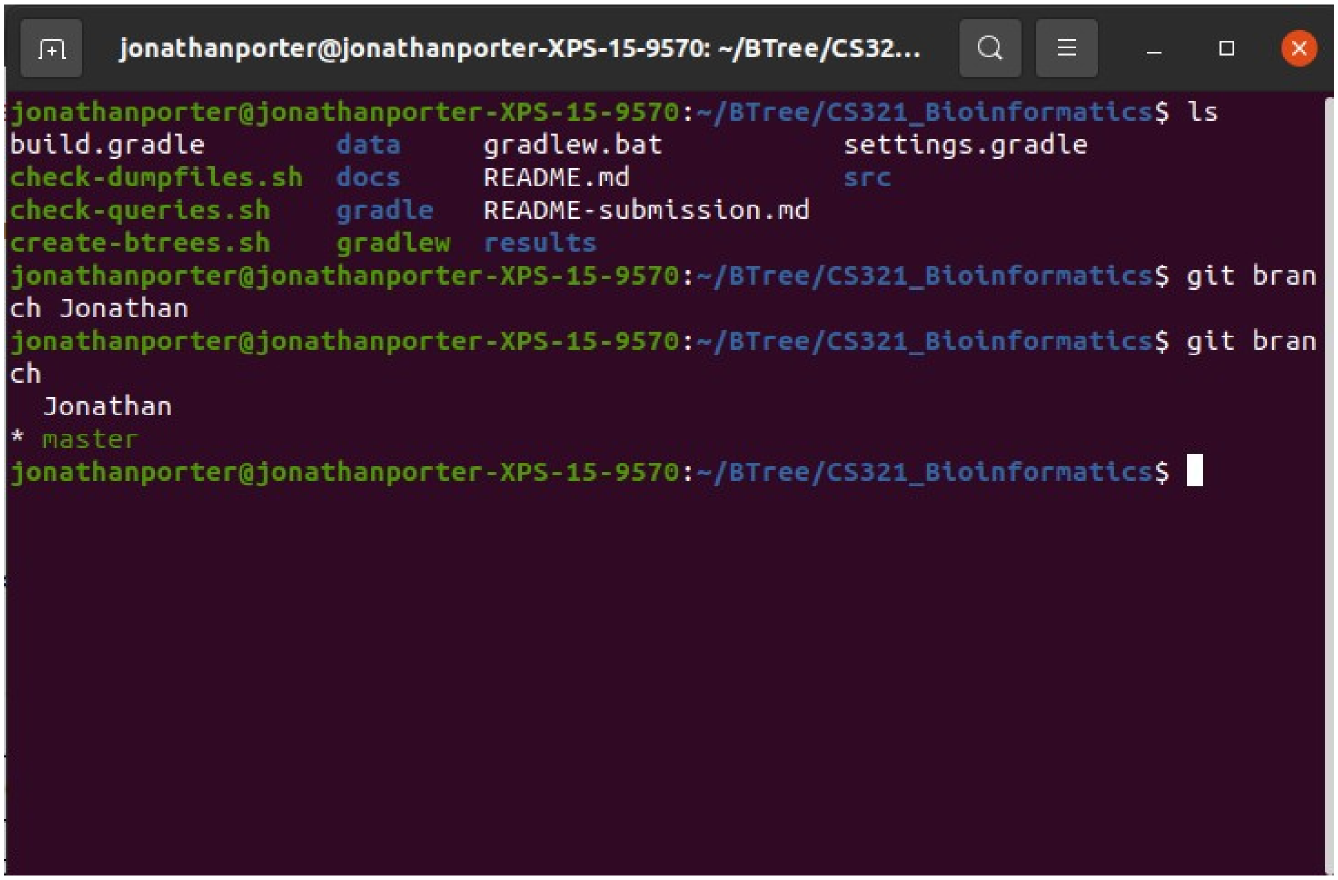
- What nodes in the B-Tree are full?.
- How is the full node in this B-Tree different from the full nodes in the previous tests?
- Add the key 13 and work out how many nodes there are, and what are the contents of each node?

Once you are passing all five of these J-Unit tests, you can now write the driver class with some assurance that your B-Tree class works.

GitHub – Getting Feedback on Code From Jonathan:

I will not be able to review everything you add into your GitHub repositories but I can see if you gave a good implementation of my suggested J-Unit tests, review code worked out in the book, see if you're on the right track with ALLOCATE-NODE(), DISK-READ(), and DISK-WRITE(). The best way you could get feedback from me is to add me as a reviewer on your pull requests. To do so, I recommend that everyone in your group works on their own branches, and merge their contributions into the main branch. While your working on your own branch, you can pull in your group member's contributions by rebasing your branch on main. Here's an example:

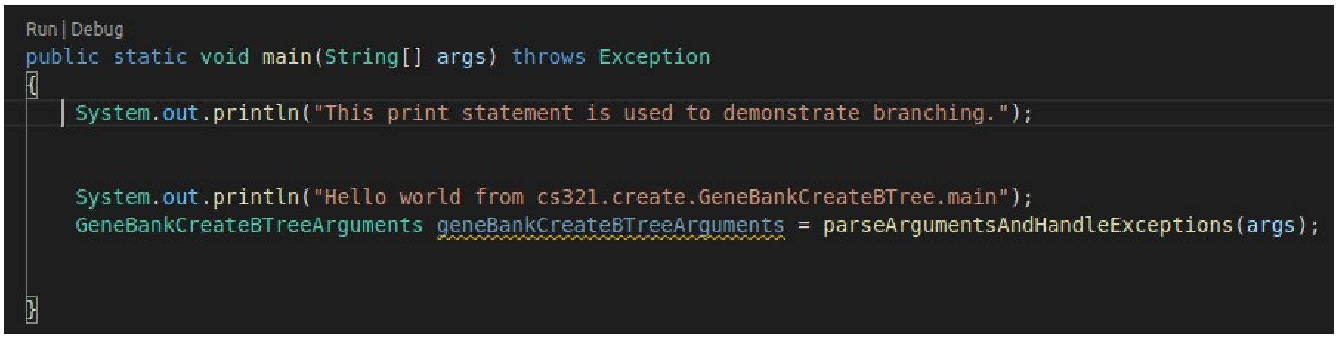
In the following image, I have cloned a new copy of the repository in the handout and I have created my own branch.



```
jonathanporter@jonathanporter-XPS-15-9570:~/BTree/CS321_Bioinformatics$ ls
build.gradle         data     gradlew.bat            settings.gradle
check-dumpfiles.sh   docs     README.md              src
check-queries.sh     gradle   README-submission.md
create-btrees.sh     gradlew  results
jonathanporter@jonathanporter-XPS-15-9570:~/BTree/CS321_Bioinformatics$ git bran
ch Jonathan
jonathanporter@jonathanporter-XPS-15-9570:~/BTree/CS321_Bioinformatics$ git bran
ch
  Jonathan
* master
jonathanporter@jonathanporter-XPS-15-9570:~/BTree/CS321_Bioinformatics$ 
```

Checkout your new branch with:
$ git checkout <branch name>

In my IDE, I've opened the cloned repository , opened the GeneBankCreateBTree class added a very simple contribution:



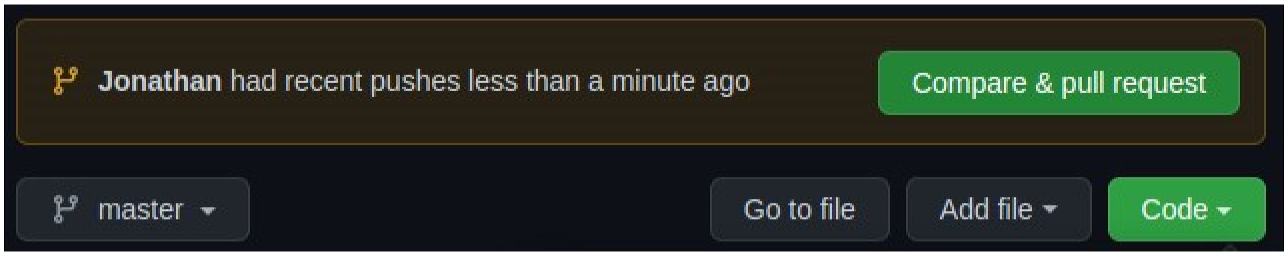
```
Run | Debug
public static void main(String[] args) throws Exception
{
    System.out.println("This print statement is used to demonstrate branching.");


    System.out.println("Hello world from cs321.create.GeneBankCreateBTree.main");
    GeneBankCreateBTreeArguments geneBankCreateBTreeArguments = parseArgumentsAndHandleExceptions(args);


}
```

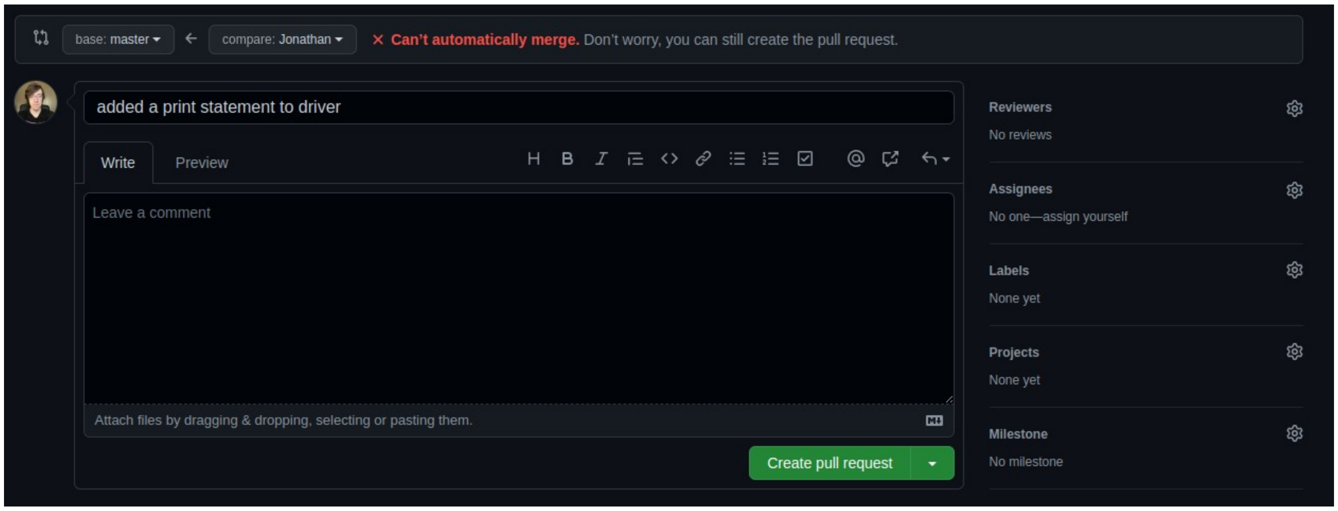
Once I save the file, I can commit and push it to GitHub via the terminal:
$ git commit -a -m “added a print statement to driver”
$ git pull -–rebase
$ git push –set-upstream origin <branch name>

Once you call Git push, GitHub will prompt you to make a pull request to merge your changes into the main branch:



Jonathan had recent pushes less than a minute ago

Click compare & pull request. Once you get to this screen:



base: master ← compare: Jonathan ✕ Can't automatically merge. Don't worry, you can still create the pull request.

added a print statement to driver

Reviewers: No reviews
Assignees: No one—assign yourself
Labels: None yet
Projects: None yet
Milestone: No milestone

Write
Preview
Leave a comment
Attach files by dragging & dropping, selecting or pasting them.
Create pull request

You can add me as a reviewer on the right and I will be able to look at the changes you’re trying to merge into main as soon as I can. Be sure to add your other group members as well, I will not review and approve the pull request unless everyone in the group is involved. Once the pull request is issued, you can still make changes to your branch. You will have to issue another pull request after you commit and push those changes to GitHub if you want them to be in the main branch. If your group members got pull requests approved from their branch and the changes were successfully merged into main, you can bring those changes into your own branch by rebasing:

```
$ git rebase
```

Your branch should now be up to date with your group members contributions.