Assignment 2: Additional Notes

Based on some queries I have received and observations I have made, below are some clarifications related to aspects of assignment 2:

1 Copy Constructor and Clone Method

It is required that you implement a copy constructor and `clone` method in the `DoubleThreadedBST` class. A copy constructor and a `clone` method both perform the same general task: they create a copy of an object belonging to the class in which they are defined. Because of this, the implementation of both will look relatively similar.

The main difference between a copy constructor and a `clone` method is what happens to the copy:

- A copy constructor acts like a normal constructor (in other words, it sets up the data in the `this` object). It receives an object parameter, which is of the same type as the `this` object (so, if you are defining a copy constructor for the `Student` class, it receives a parameter of type `Student`). The copy constructor then sets up the data in the `this` object to be a copy of the parameter object.

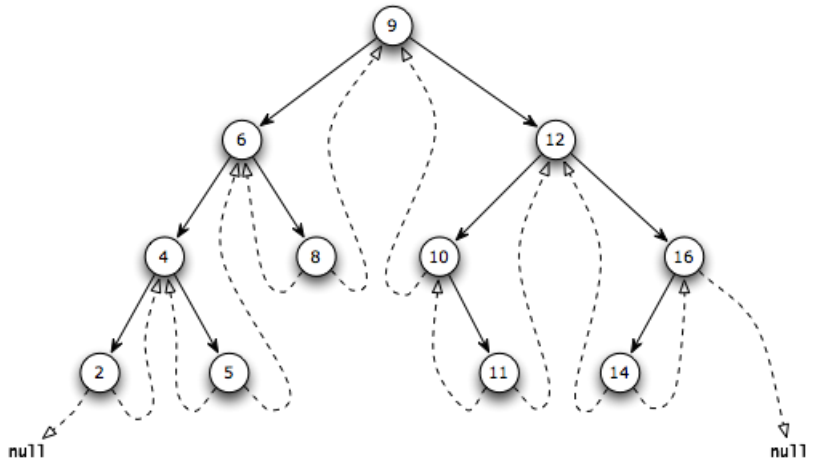
- A `clone` method creates a new object and returns it. Therefore, if we are defining a `clone` method in the `Student` class, it creates a new object of type `Student`. This new object is created as a copy of the `this` object, and is then returned.

The most important thing to remember is that the copy created by both a copy constructor and a `clone` method should be a *deep copy*. This means that it must be a copy of both the `Student` object, as well as all the data stored in the `Student` object. So, if a `Student` object contains a `Parent` object, the copy constructor and `clone` method in `Student` must also make a copy of the `Parent` object.

The simplest way to test whether your copy constructor and `clone` method are working correctly is to use them to make a copy of an existing object, then change data in the original object, and ensure that the data in the copy has not been changed. In the case of the assignment, let us assume that we have a `DoubleThreadedBST` object called `obj`. Also assume you create a copy of `obj` using either the copy constructor or the `clone` method, and call this copy `cpy`. Now, if you change `obj`, you should observe that `cpy` does not change (the opposite should also be true). If you change `obj` and `cpy` does change, a shallow copy is being created, and this is an error.

2 The Pipe Symbols in `inorderReverseDetailed` and `preorderDetailed`

Consider the following diagram of a double threaded BST:



Note that a pipe symbol is printed whenever a right or left thread is followed to another node in the traversal. This means that multiple pipe symbols may be printed next to each other if a chain of threads is followed higher up the tree. A pipe symbol is not printed when a thread is followed to a `null` value, which typically stops the traversal.

For example, consider the diagram above, and assume that your traversal is currently printing node 11. Assume that you then follow the left thread from node 11 up to node 10, don't print node 10, then follow the left thread from node 10 to node 9, and finally print node 9. Your traversal would then print `11||9`. This is because you've not printed node 10, but have followed two threads from node 10 to get to node 9, which is then printed.

It is also possible that a traversal may end with one or more pipe symbols. This happens if you follow one or more threads, but don't print any of the nodes that you reach, and then end your traversal.

Also note that traversals using a threaded tree typically end when a node is reached from which you would follow a thread to a null value (again, look at the figure above). Note that in this case you must not print a pipe when following a thread to a `null` value, because this ends the traversal.

3 Details Related to Tree Operations

Pay careful attention to the descriptions of the methods in the provided code. In particular note that the deletion is the opposite of the approach described in the textbook and slides. Also note that the inorder traversal is a reverse traversal, while the preorder traversal is the standard traversal discussed in the textbook and slides. When in doubt, work through the operations on paper before you begin implementing program code.