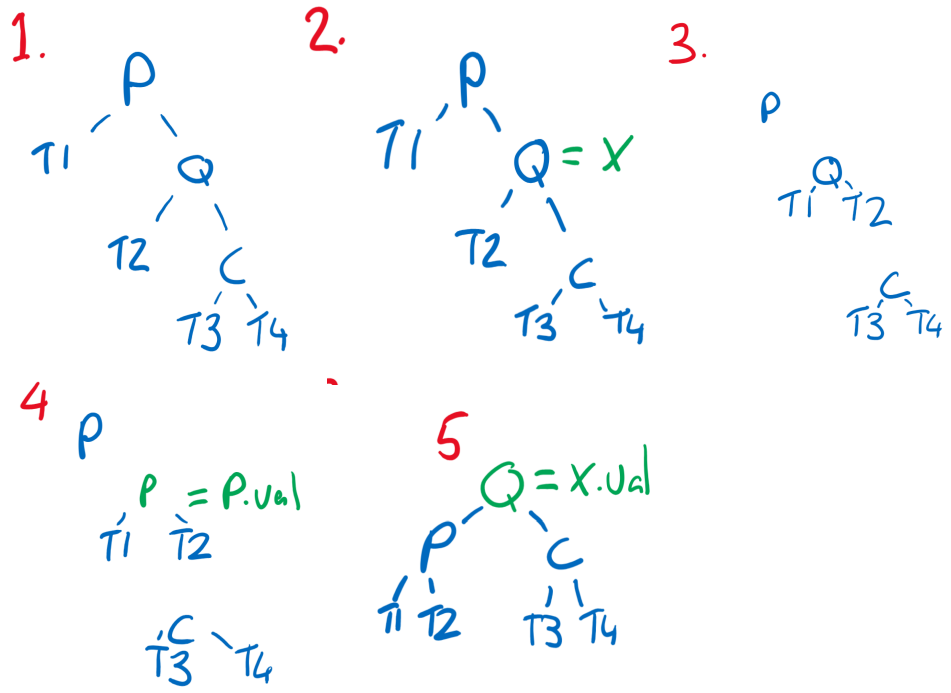
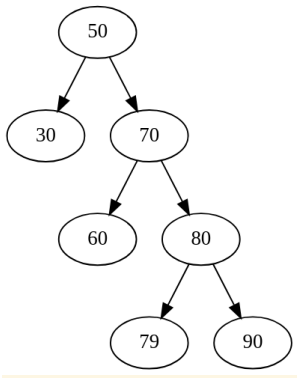
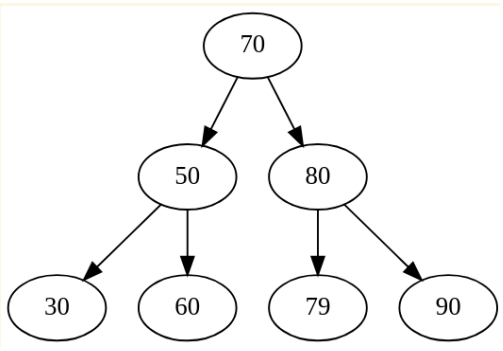
Report:

1. This has been done
2. Found first common ancestor by searching for a value then adding each step to a list. When both values are found walk through both lists for the first same value
3. I want to briefly describe my approach. Since In the code we didn't make use of the parent field in the node definition I had decided to work with changing the values of each node. Although in just changing the internal fields I did not change the names of each field. So I had followed the “yoink” technique but it was using the internal fields rather than the whole node.

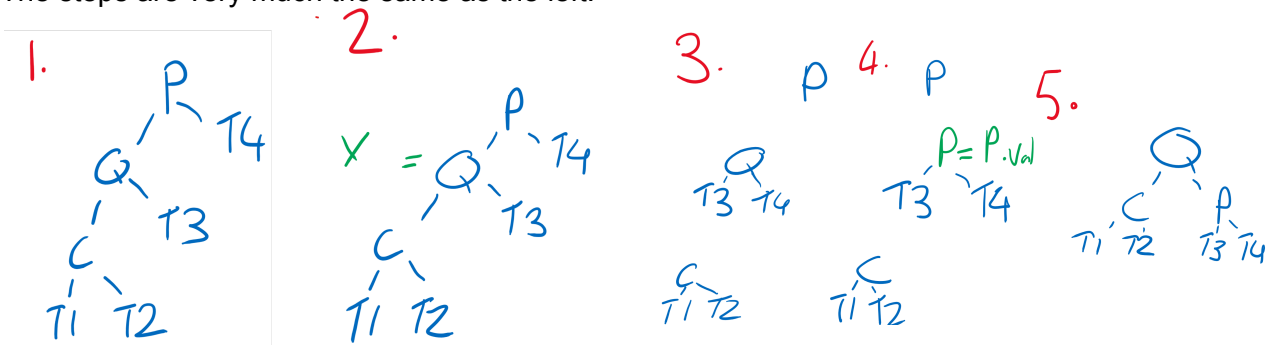


Left:
So the steps
The initial step is to find the variables P, Q and C.
Then I take the value of Q and put it in a temporary variable X.
The third step is to put the left and right value of Q to T1 and T2.
The fourth is to make Q the value of P functionally making it P.
The fifth step is to make P into Q by giving it Q’s value (X as the temporary value). P and Q are then put into Q’s left and right values.

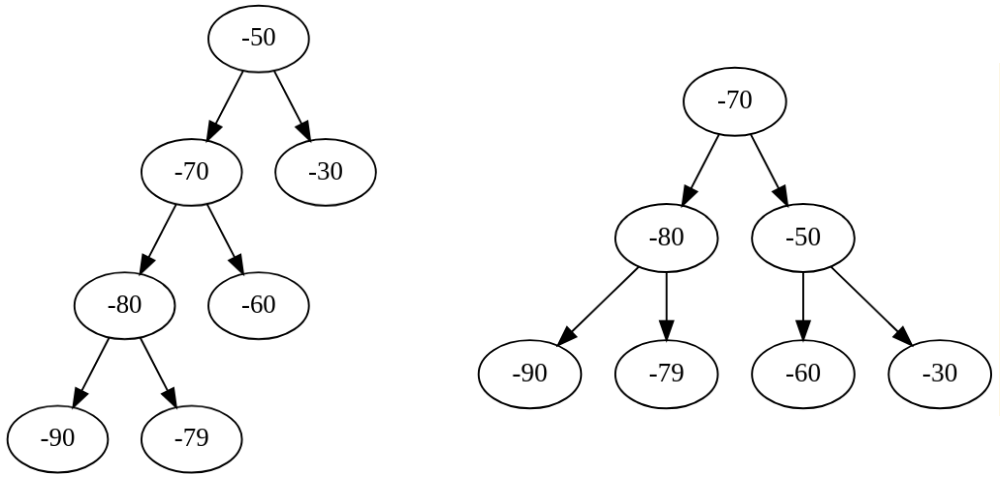




Right:
So the steps
The steps are very much the same as the left.

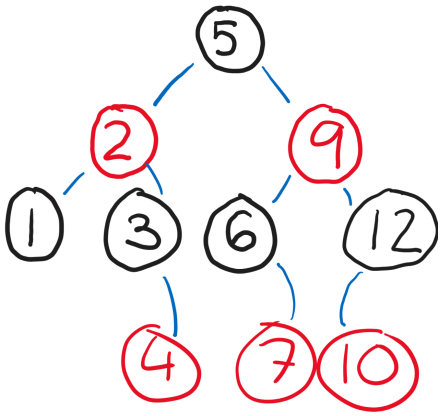


And the results can be seen below:

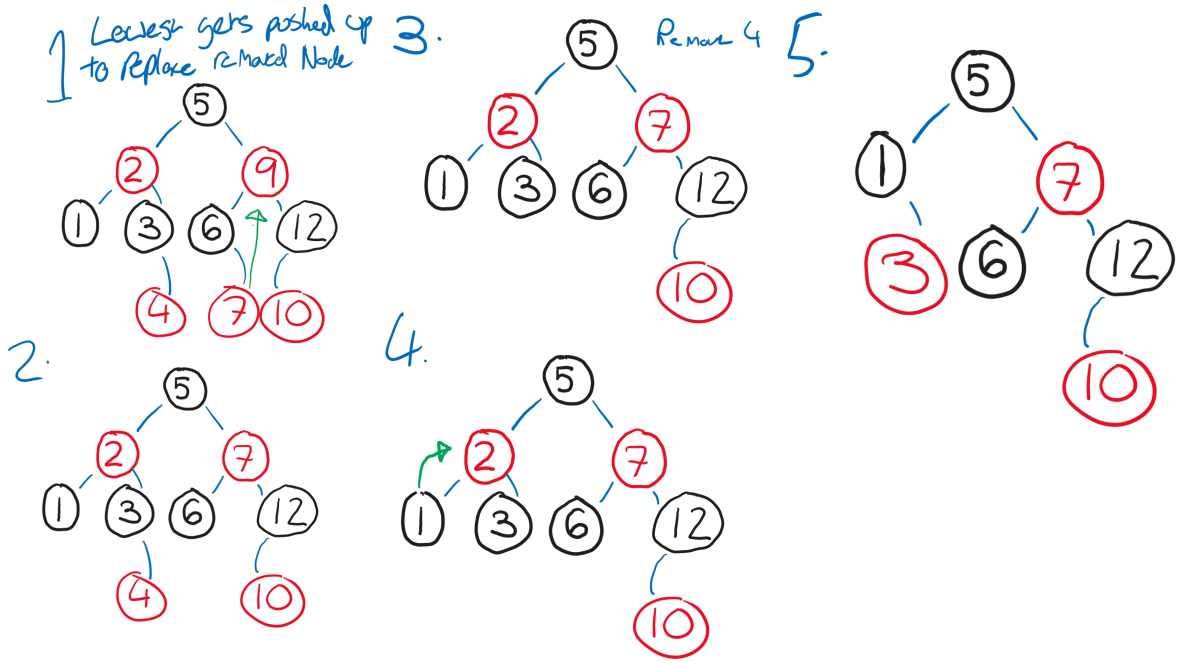


The left-right and right-left operate the same just by joining the two functions.

4. Inserted the values:

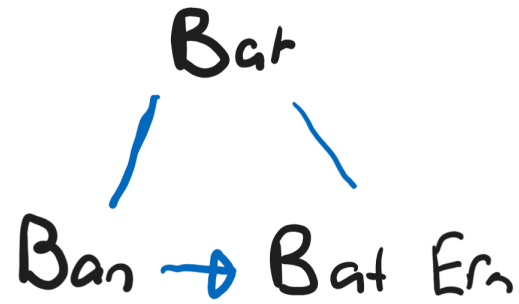


Removing Values:

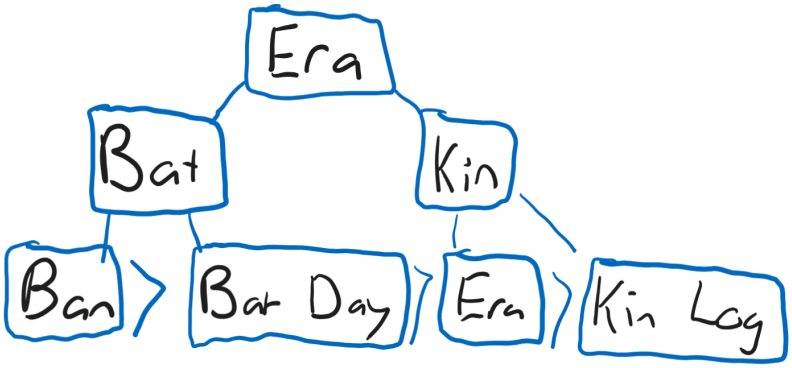


5. Each of the steps are as follows:

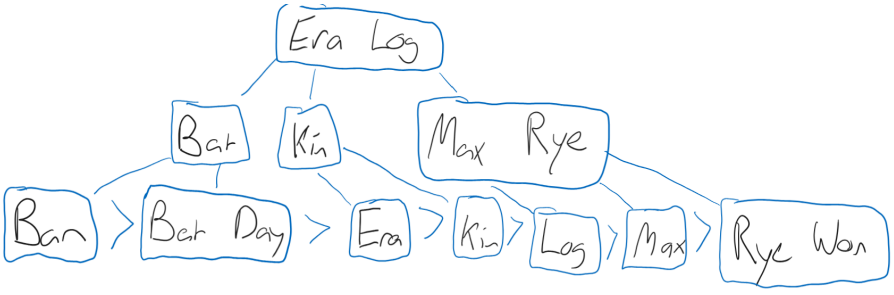
a.



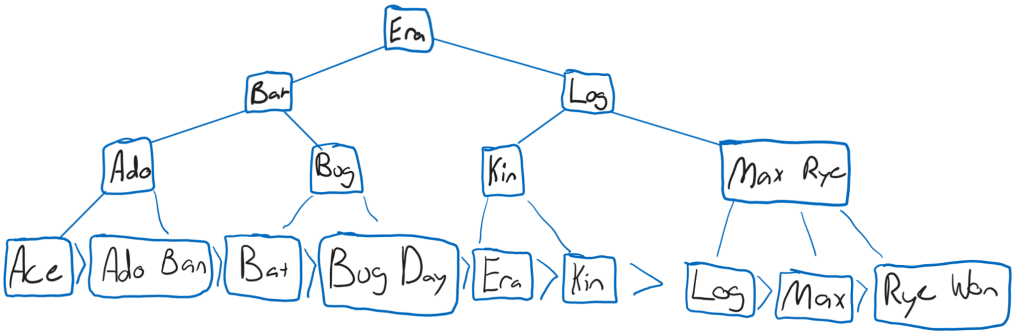
b.



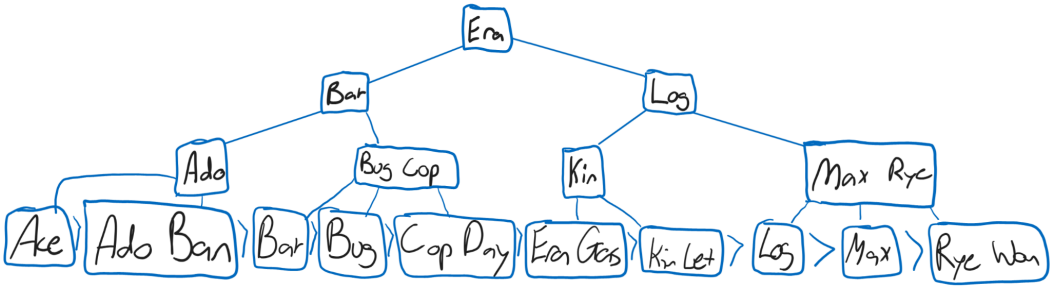
c.



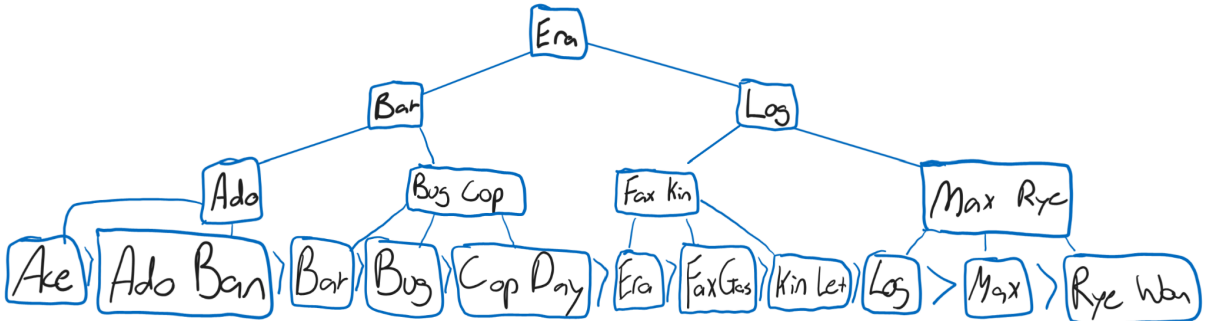
d.



e.



6. Final Graph:



Module Reflection

A good reflection will include statements like:

This module had really extended my knowledge about trees. It was very interesting to look at how we can store data in a method that can be very effective. I did really like the coverage of how there can be different tree constructions like RB and AVL trees. Although I did find the B+ graph super interesting since it is so different to the other graphs we learnt about. Although I do wish it had some more coverage. I also did find working with the code super valuable since performing the rotations seemed very similar to working with pointers in languages like C and it did lead me to learn much more about how objects work with python. I am also keen to discuss the code with my tutor since I am a bit concerned there could be things to improve upon or see other methods for solving the same problems.

Code reflection

I did find the code has not utilised the parent field in the node object. I did think that made the task harder since that is a part of the code that could be helpful. I also found the code at times hard to understand since it was very efficient and sometimes the comments did not describe what was going on enough.