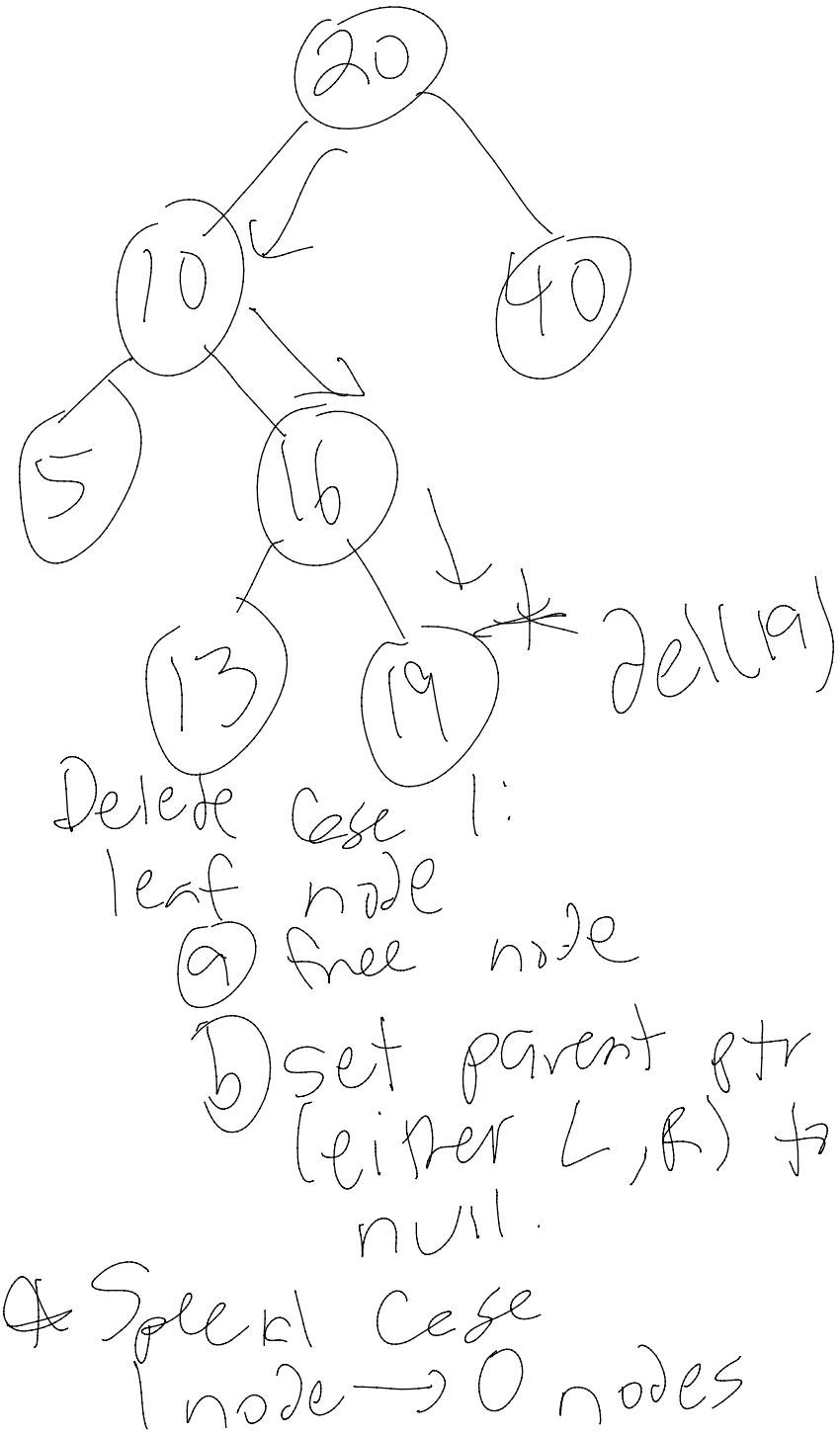
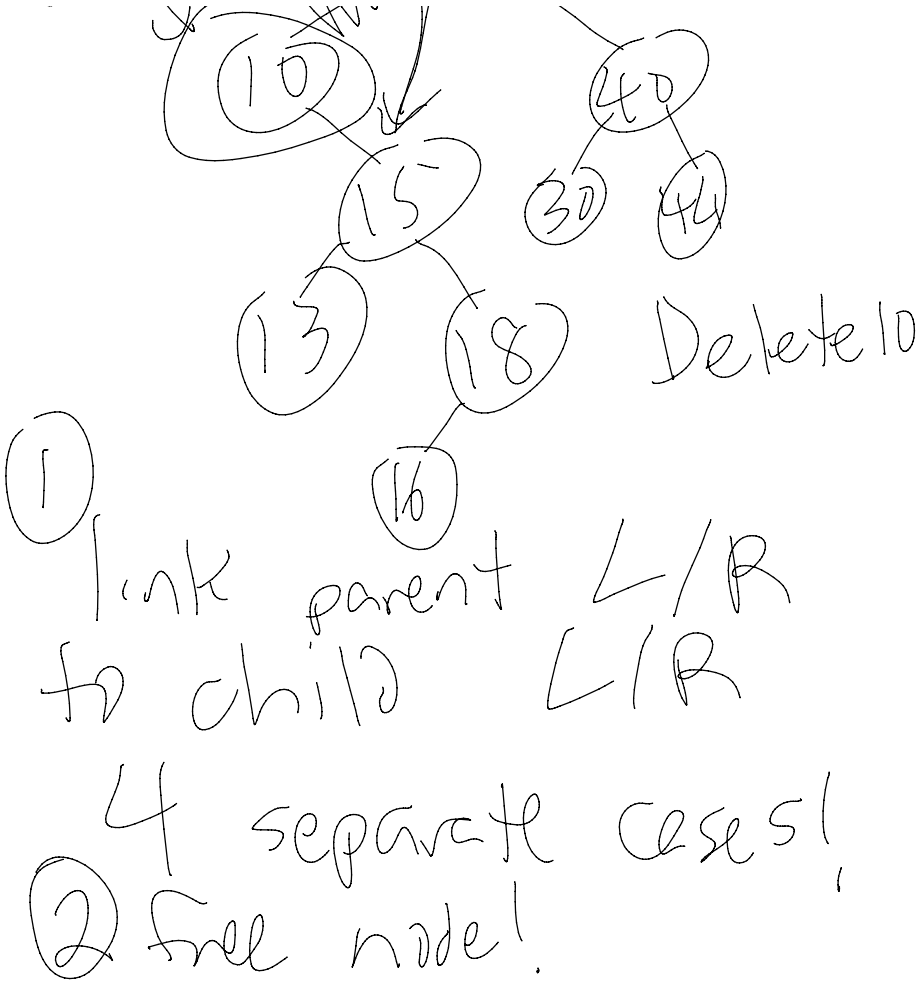
# 7/6/2020 - Binary Search Tree Delete, Practice Bin Tree Questions

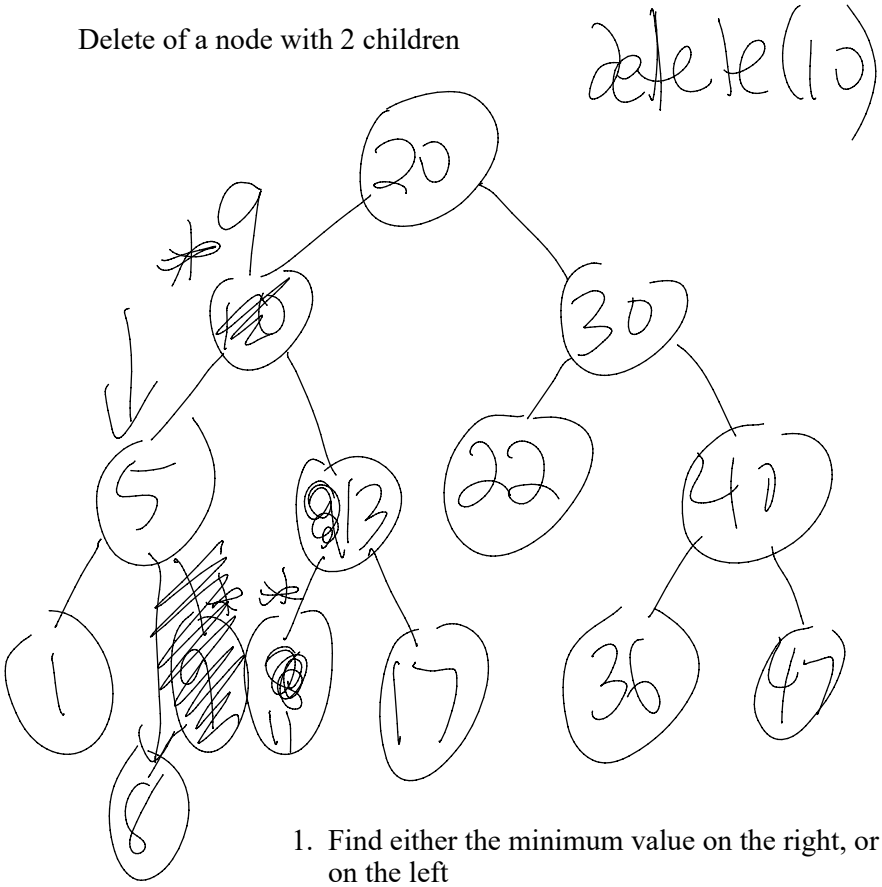
Monday, July 6, 2020 3:58 PM




Next case: node to delete has 1 child



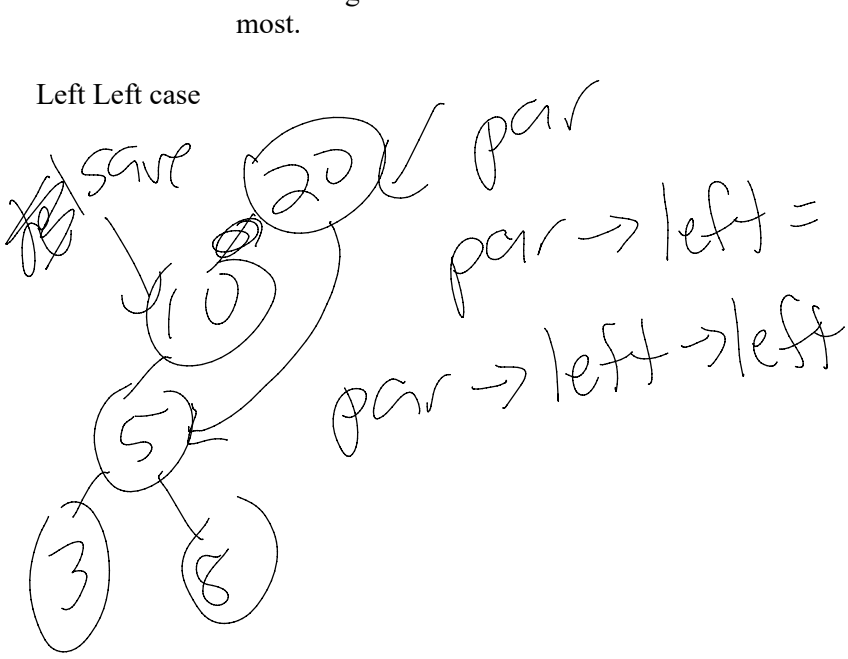

Delete of a node with 2 children

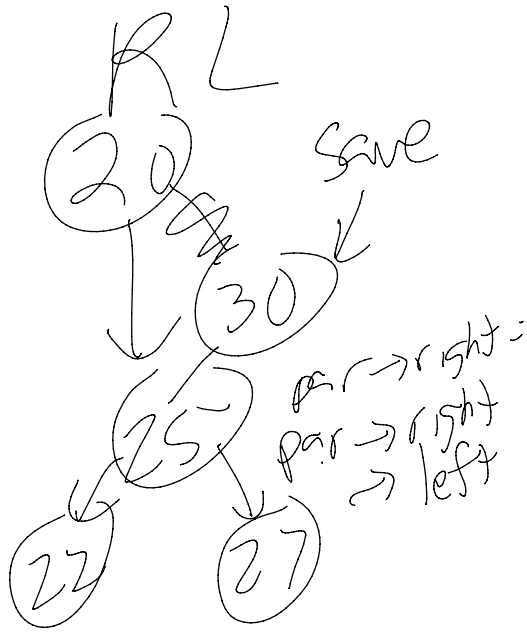



1. Find either the minimum value on the right, or the maximum value on the left
2. Copy this value into where we would have done our delete, and then

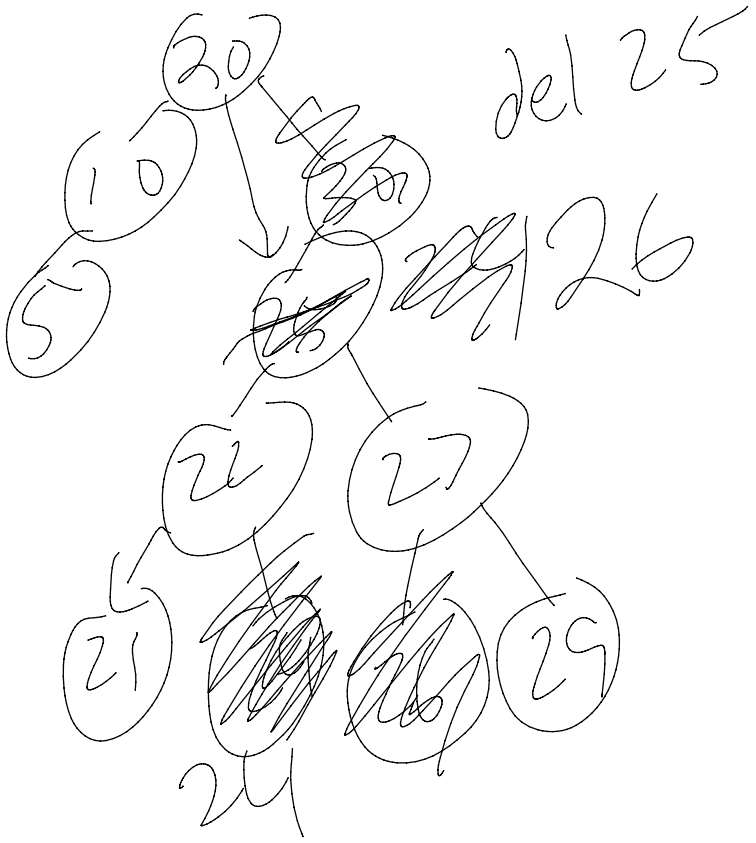
delete the physical that used to store the value.
Max on left can not have a right child, since if it did, it wouldn't be the max…
min on the left can not have a left child, since if it did, it would not be the min…
So we're guaranteed that the new node we delete ahd only 1 child at most.

Left Left case




Class Trace Tree

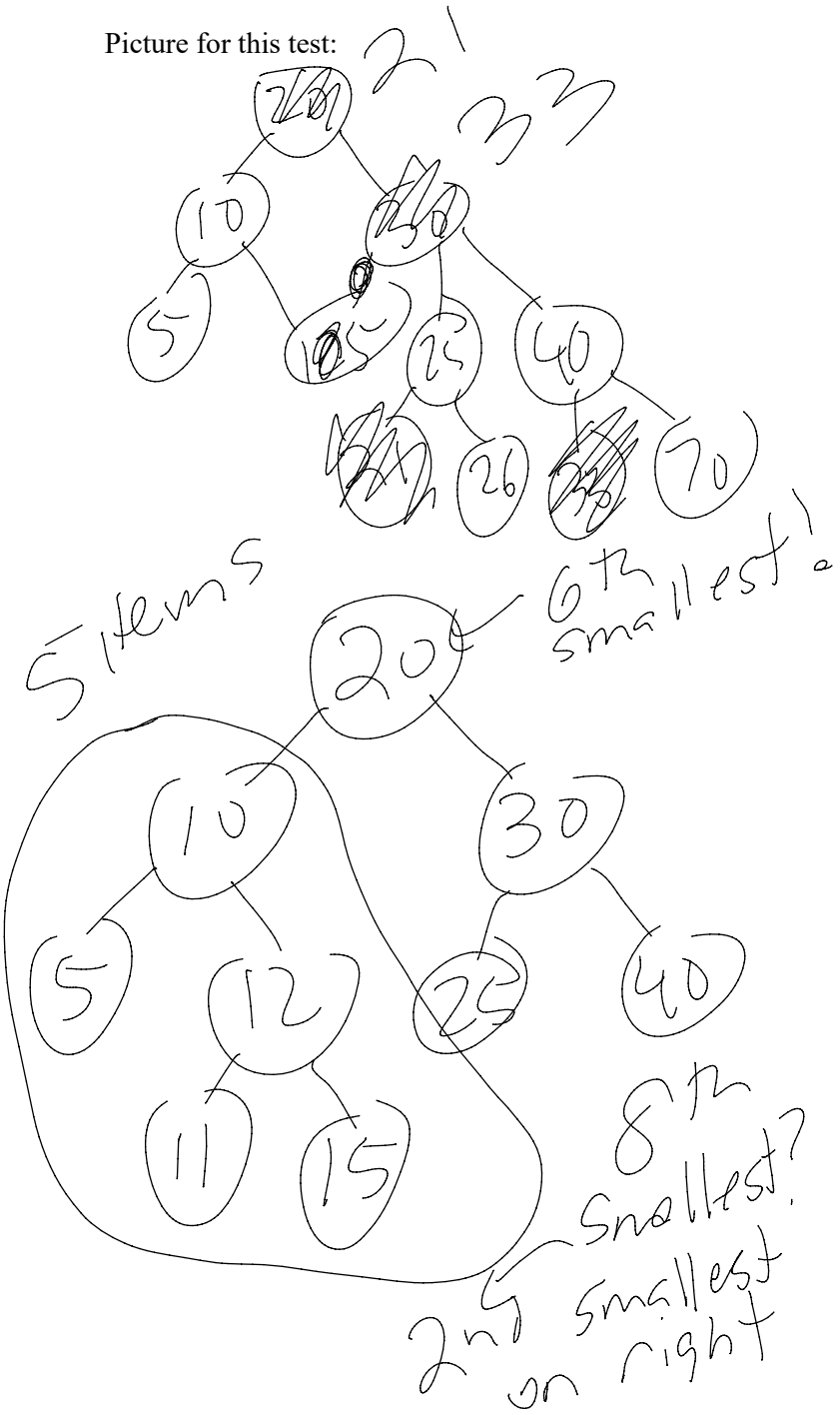



Recursive Delete Structure:

if the item is the node you are deleting…
    check if you have 0, 1 or 2 children.
    if 0 children, free node, return NULL.
    if 1 child, save a ptr to the child,
    free your node, and then return child ptr
    if 2 children,
        delete the min val on right
        copy this value over into the node itself
        return the node itself.

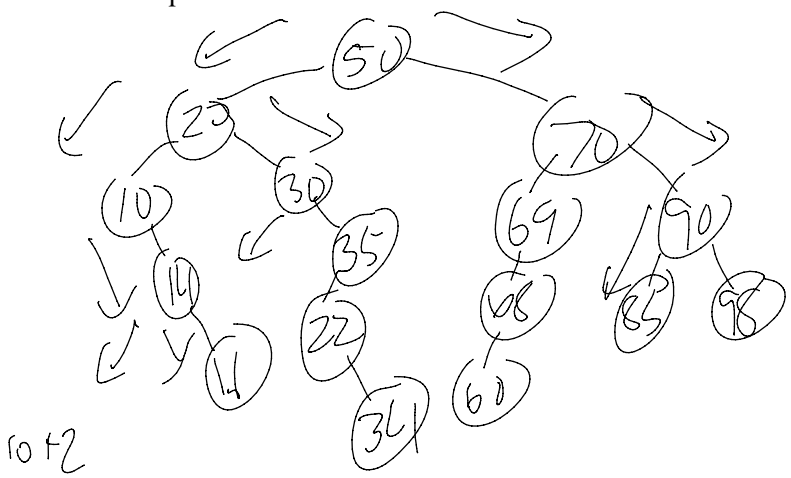
if you are on the left, recursively delete the node in the left child and reassign left.
OR
recursively, delete the node in the right child and reassign right.

Picture for this test:

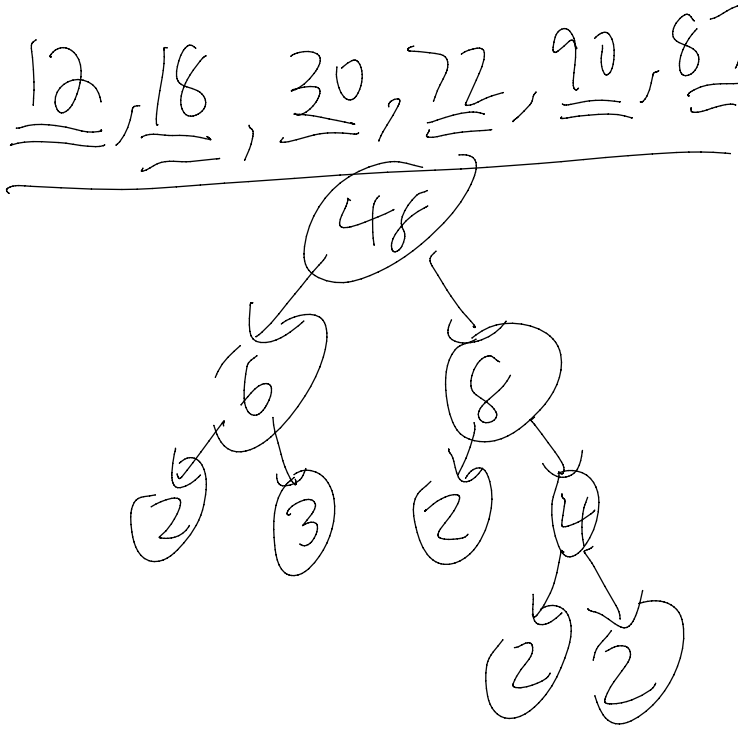


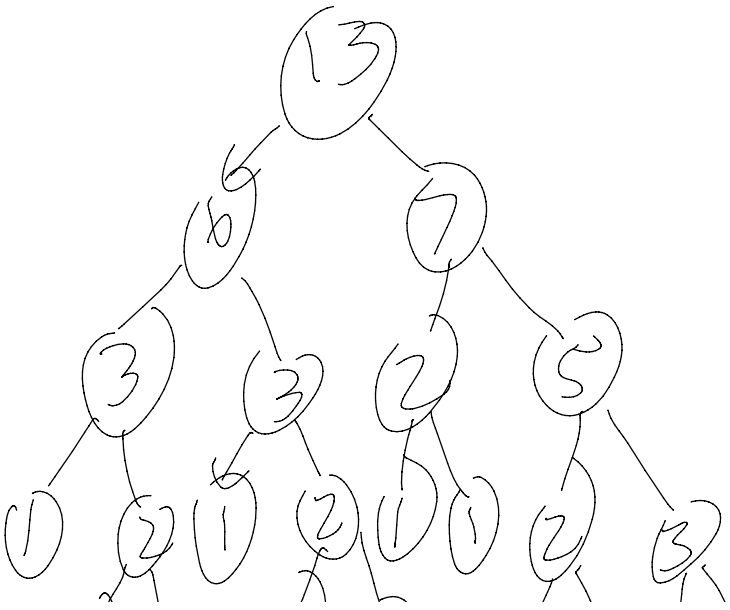
Tips for binary tree traces…

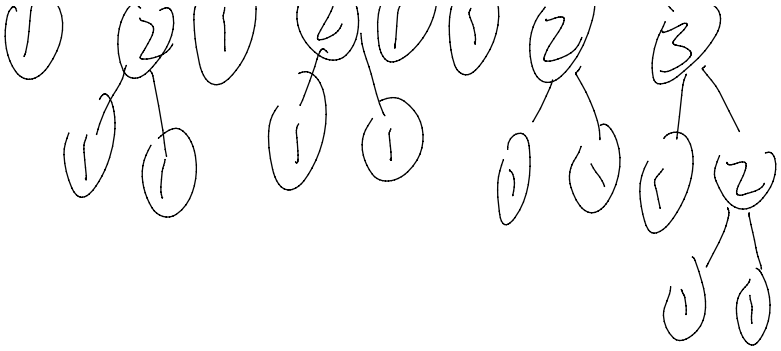
First question on the notes



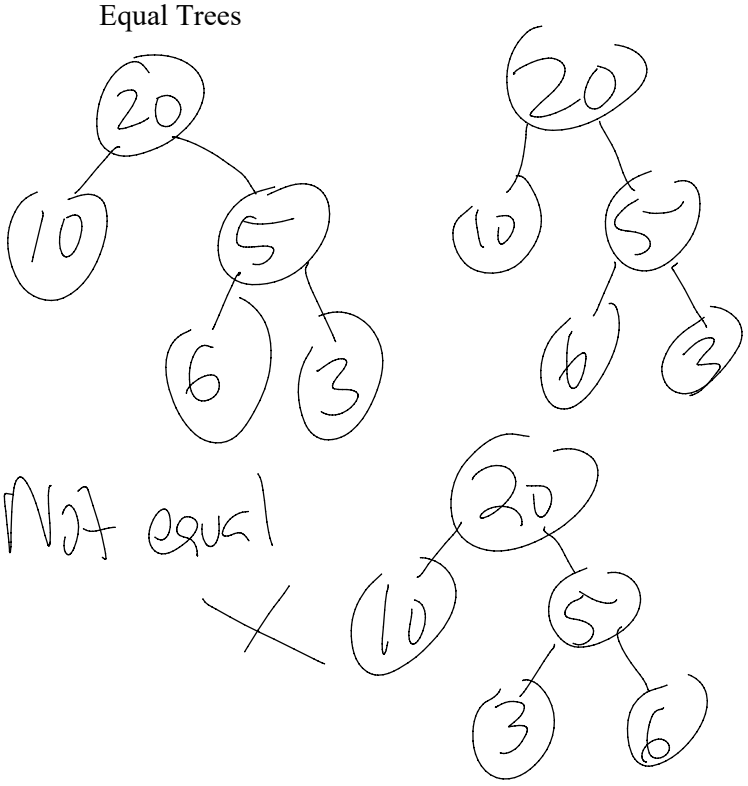

10 42





Equal Trees




Base cases:

both trees NULL - return 1
if only 1 tree is NULL - return 0
if the roots of the two trees are not equal - return 0
if roots are equal-
recursively return equal(A->left, B->left) &&
equal(A->right, B->right)

This is written "opposite" of the typed up notes…

My solution to the one child problem:

```
int numNodesOneChild(struct tree_node* root) {
   if (root == NULL) return 0;
   int res = 0;
   if (root->left == NULL && root->right != NULL)
      res = 1;
   else if (root->left != NULL && root->right == NULL)
      res = 1;
   return res + numNodesOne(root->left) +
                numNodesOne(root->right);
}
```

```
int numNodesOneChild(struct tree_node* root) {
   if (root == NULL) return 0;
   int res = 0;
   if (root->left == NULL && root->right != NULL)
      return 1 + numNodesOneChild(root->right);
   else if (root->left != NULL && root->right == NULL)
      return 1 + numNodesOneChild(root->left);
   else
      return  numNodesOneChild(root->left) +
                numNodesOneChild(root->right);
}

void printPassword(treenode* root) {
   if (root != NULL) {
      printPassword(root->left);
      if (isConsonant(root->ch))
         printf("%c", root->ch);
      printPassword(root->right);
   }
}
```