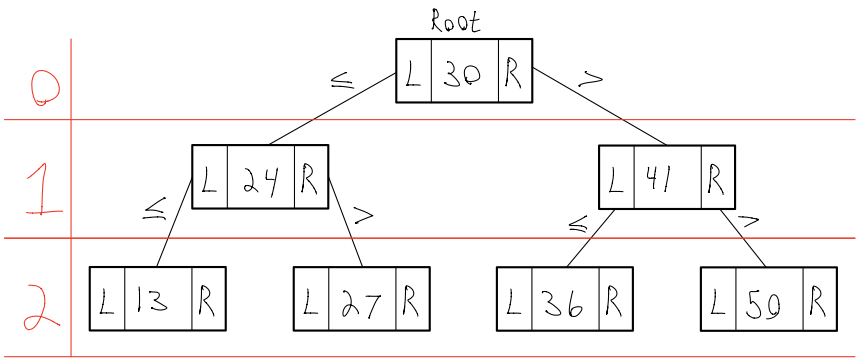
Any code with a green square around it is what I actually implemented into the assignment. Everything else is just a rough draft

L = left Pointer
R = right Pointer
Key = node->data



Add
Subtract
display

add Subtract

Traversal:
if input ≤ Key:
go Left
elif input > Key:
go right
else:
Found input

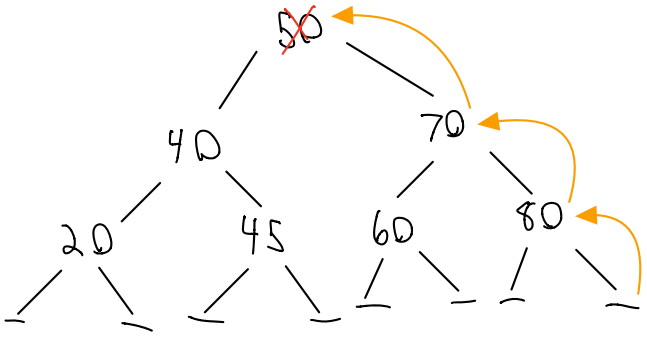
(returns if node = NULL)

New node:
use when adding to list

~~Create(value)~~
~~*node = new node~~
~~node->value = value~~
~~node->left = nullptr~~
~~node->right = nullptr~~
~~return node~~

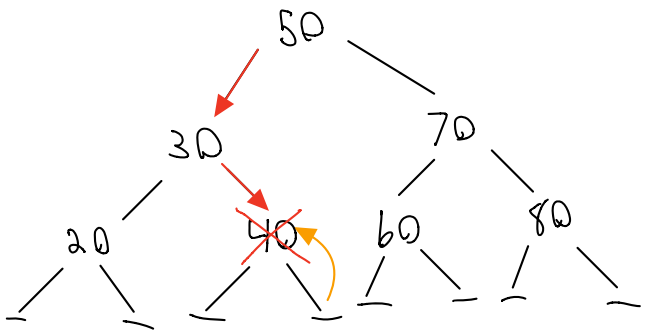
```
add(node, value)
  if node = nullptr:
    ~~return Create(value)~~
    return new BSTnode(value)
  elif value ≤ node->value:
    node->left = add(node->left, value)
  else:
    node->right = add(node->right, value)
```

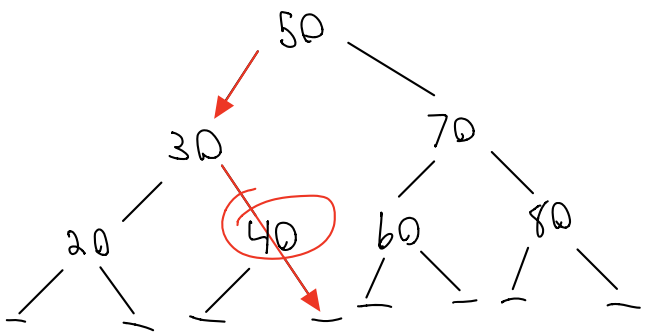
Remove



remove 40

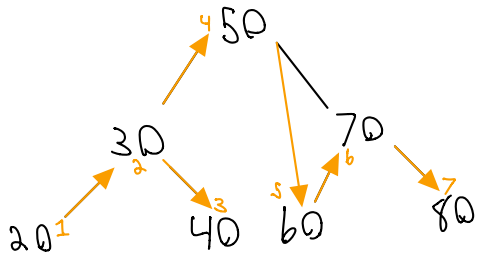
```
If value < node->value:
  remove(node->left, value)
elif value > node->value:
  remove(node->right, value)
else:
  node->value = node->right->value
```





Traversal (inorder)

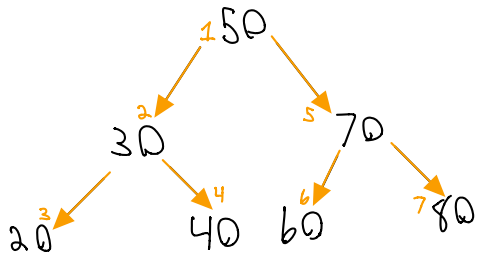
```
display(root):
if root != Null:
    display(root->left)   #Stops when root->left = NuLL
    Print(root->value)    # then prints value of leftmost
    display(root->right)  #Same as left but for right
```



preorder & postorder should be similar to inorder just mixed up a little

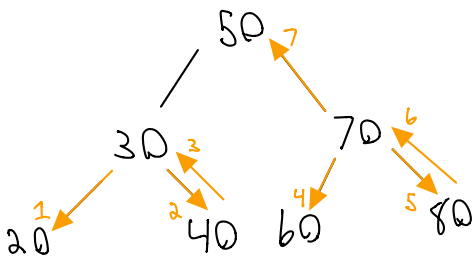
Traversal (Preorder)

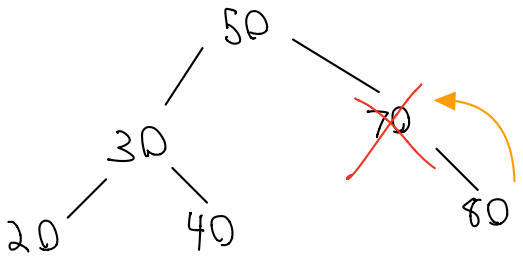
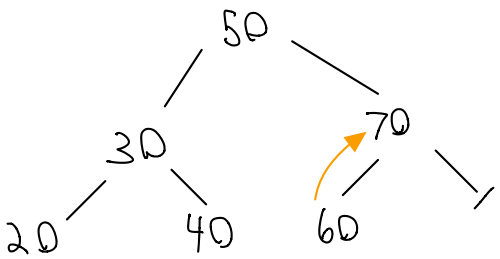
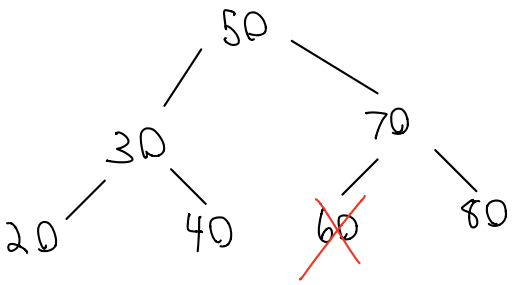
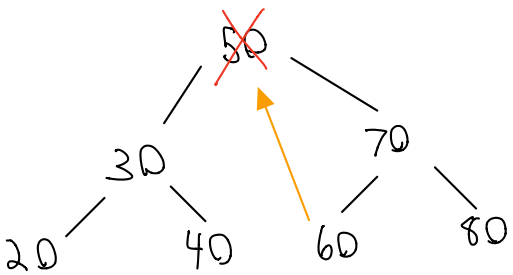
```
display(root):
if root != Null:
    Print(root)
    display(root->left)
    display(root->right)
```



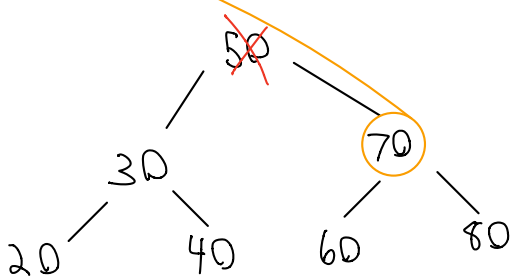
Traversal (Postorder)

```
display(root):
if root != Null:
    display(root->left)
    display(root->right)
    Print(root)
```



only one child (right)		copy 80 to 70 then delete 80
only one child (left)		copy 60 to 70 then delete 60
No child nodes		Only need to delete 60
Two child nodes		Find inorder successor then copy to 50 and delete successor Allows us to not have to shift an entire branch up

```
Min(node)   # First Pass it root->right
BSTnode *current = node
While current->left != NULL:
    current = current->left
return current
```



```
delete(node, value)
if value < node:
    delete(node->left, value)
elif value > node:
    delete(node->right, value)
else:
    if node->left == NULL:
        BSTnode *temp = node->left
        delete(node)
        return temp
    elif node->right == NULL:
        BSTnode *temp = node->right
        delete(node)
        return temp

    BSTnode *temp = min(node->right)
    node->value = temp->value
    node->right = delete(node->right, temp->value)
```