int add (node * & root, video & toadd) {
    if (root == NULL) {
        root = new node;
        root -> data = copy(toadd);
        root -> left = root -> right = NULL;
    } else if (root -> data.compare(toadd) < 0) {
        // if root's data less data to add
        return add(root -> right, toadd);
    } else {
        return add(root -> left, toadd);
    }
}
(C ≠ Java) ← useful

c++

```cpp
node * table::add ( node * root,
    video & toadd )
{
    if (root == nullptr)
    {
        root = new node;
        root->data = copy ( toadd );
        root->left = root->right = NULL;
        return root;
    }
    else if ( root->data.compare ( toadd ) < 0 )
    {
        root->right = add ( root->right, toadd );
    }
    else
    {
        root->left = add ( root->left, toadd );
    }
    return root;
}
```
C++

```
int table::add (node **proot, video &toadd)
{
    if (*proot == NULL)
    {
        *proot = new node;
        (*proot)->data = toadd;
        (*proot)->left = (*proot)->right = NULL;
    }
    else if (/* some condition */) {
        return add(&(*proot)->left, toadd);
    }
```

INTERNAL NODE 2 children

FIND IN-ORDER SUCCESSOR

1. Go right one
2. Spin left until current->left is NULL
3. Copy the data!

4. //adoption!
   prev->left = current->right

5. delete current's node

Delete 140?

(1) * check to make sure the right child has a left first

(NO) you have no successor!
Remove

1. Tree is Empty, Nothing to Remove

2. The Item to Remove cannot be found.

3. Leaf (No children)
   - Both left & right pointers are NULL
   - delete the node
   - set root to NULL

7. One Child (left)
   - left is NOT NULL, right is NULL
   - hold onto your left child
   - delete the node
   - set root to left child
5. One child (right)
   - Left is null, Right is NOT NULL
   - Hold onto your right child
   - Delete the node
   - Set root to right child

6. Internal node w/ 2 children
   - Both left & right are not null
   - Go right once
   - Check if Right's left is null
     - YES: The "Right" is median
       - a) Copy data into our node
       - b) Adopt "Right"'s right
       - c) Delete "Right" child
     - NO:
       - a) Churn left until current->left is null
       - b) Drag previous pointer
       - c) Copy current's data to our node
       - d) previous->left adopt current's right
e) delete node that current is pointing at