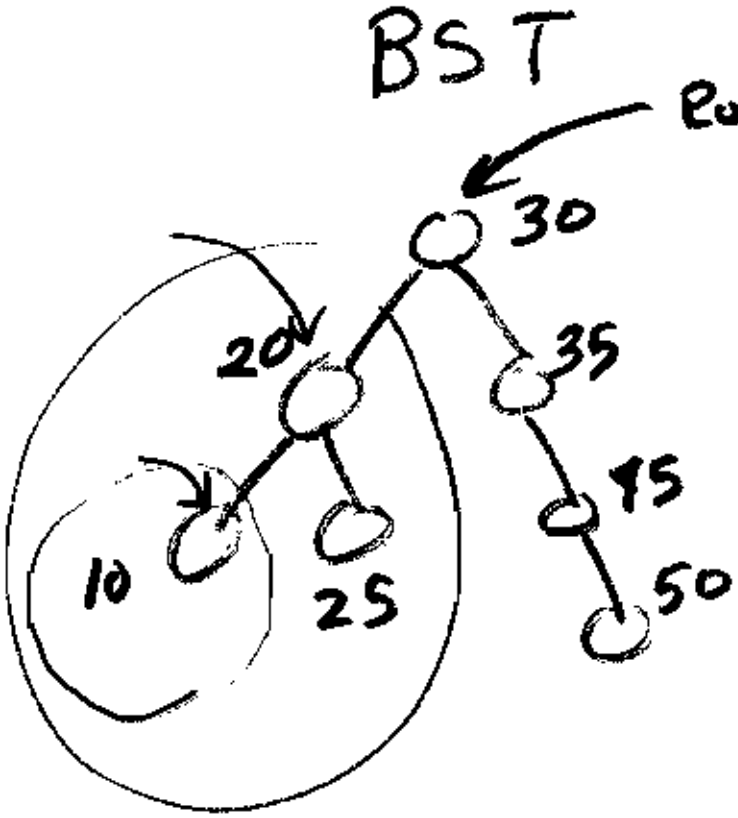



```
int tabb::add(node *& root, video & toadd)
{
  if(root == NULL)
  {
    root = new node;
    root->data.copy(toadd);
    root->left = root->right = NULL;
  }
  else if(root->data.compare(toadd) < 0)
    // if root's data less data to add
    return add(root->right, toadd);
  else return add(root->left,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

if(!root) return 0;

return 1; return 2;

(C & Java) ← useful

C++

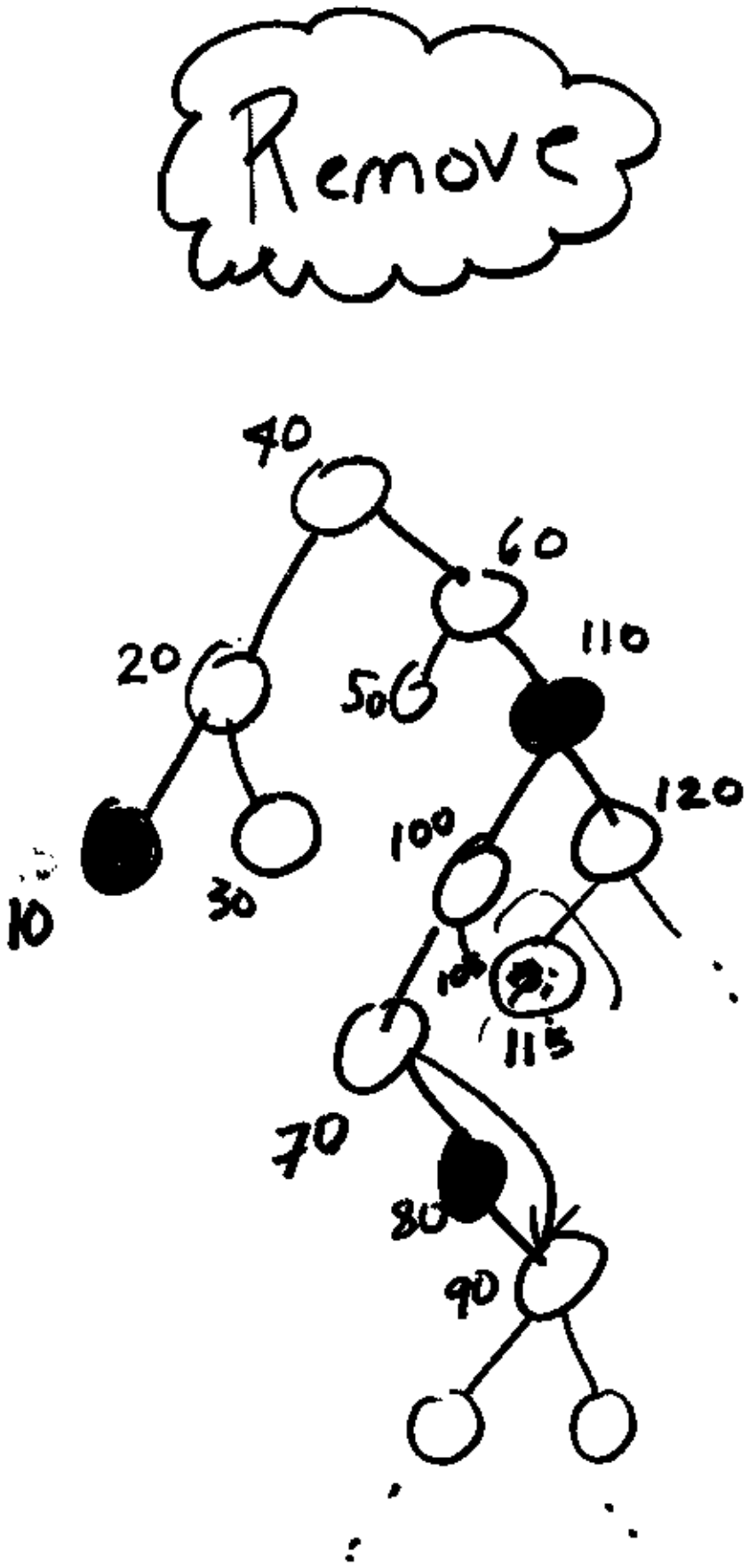
```
node *  table::add( node *root,
                    video & toadd )

{   if (!root)
    {   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)

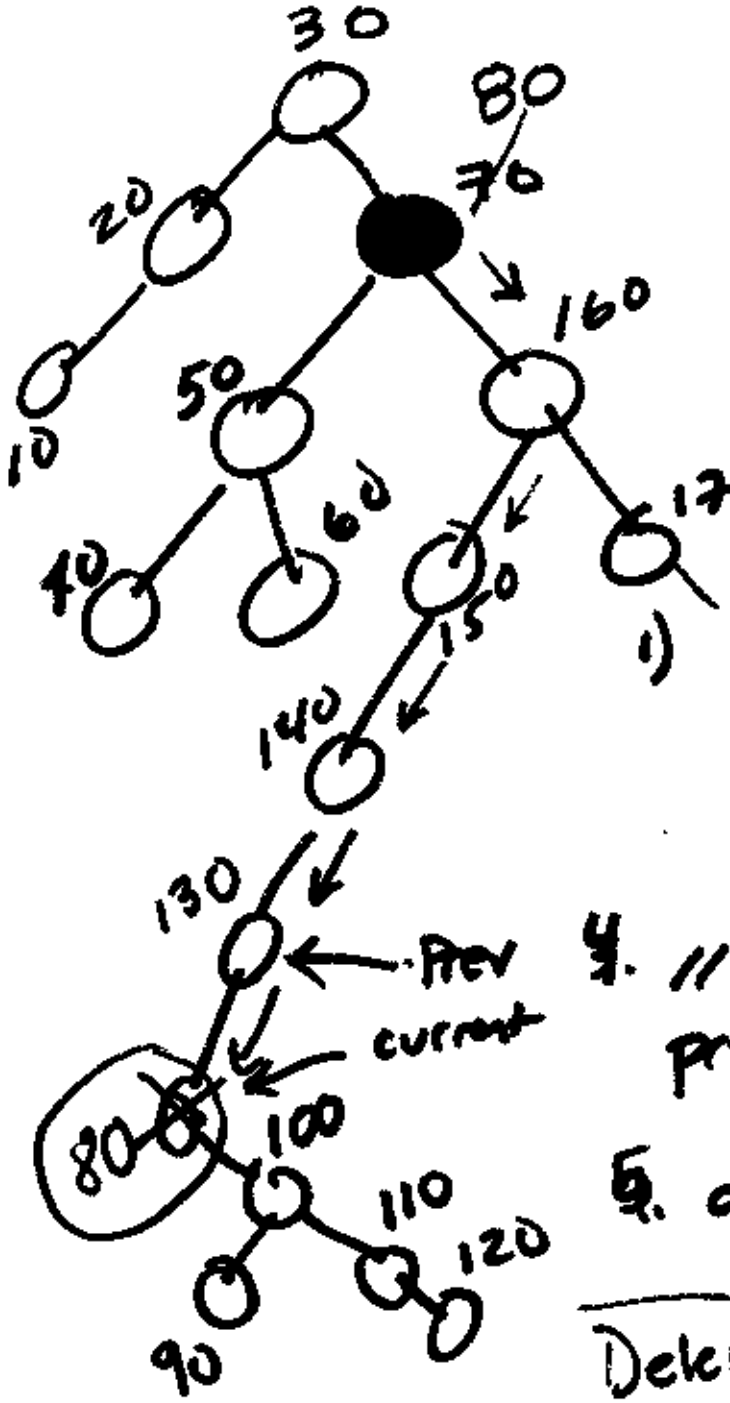
C++

pass by pointer

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


Remove
40
60
20
50
110
10
30
100
120
115
70
80
90

INTERNAL NODE 2 children



FIND IN-ORDER SUCCESSOR

1. Go right once
2. Spin left until current->left is NULL
3. copy the data!
4. //adoption!
   prev->left = current->right
5. delete current's node

Delete 160?

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

NO ← you have inorder successor!

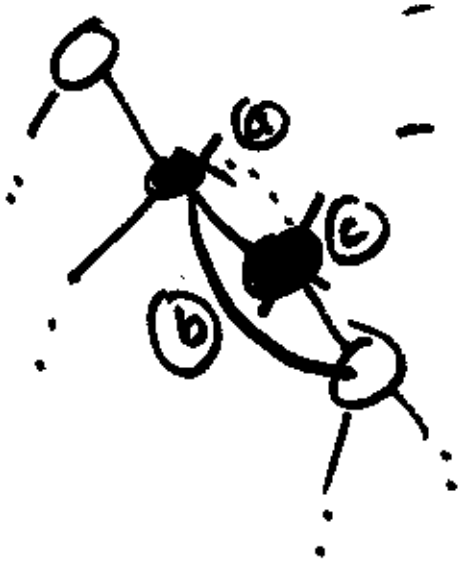
Remove

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
4. One Child (left)
   - Left IS NOT NULL, Right is NULL
   - hold onto your left child
   - delete the node
   - set root to left child

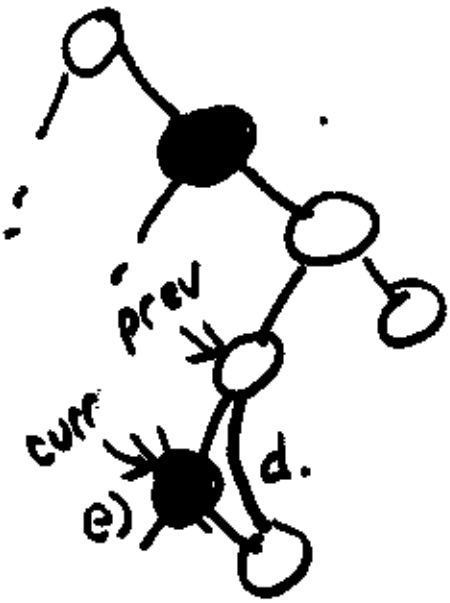
5. One child (right)

- 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 inorder successor
    a) copy data into our Node
    b) Adopt "Right's" right child
    c) delete "Right"



- 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