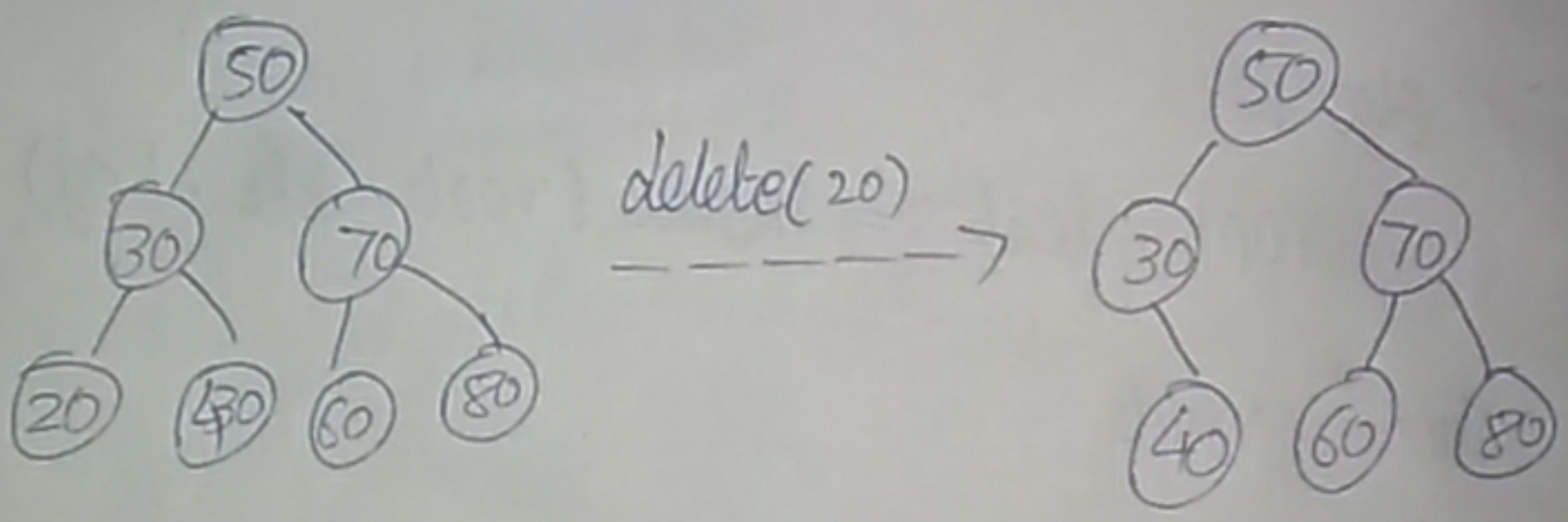
Day 24

Deletion in binary Search tree:

when we delete a node, three posibilities arise.

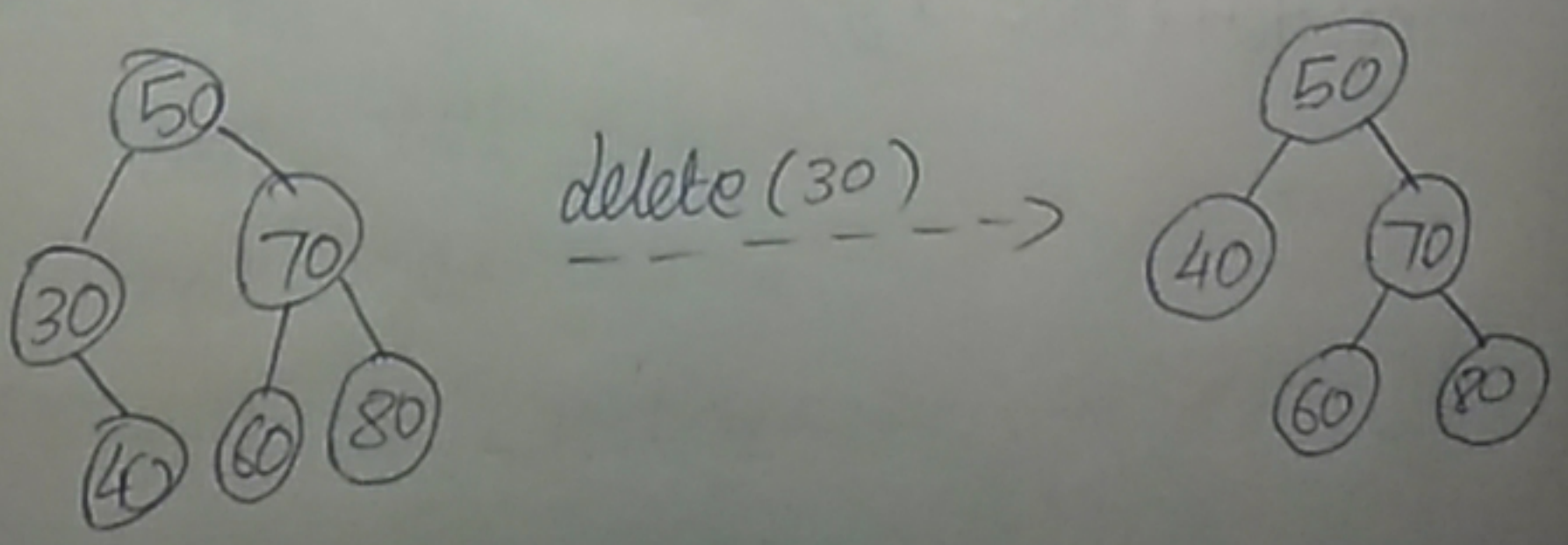
Node to be deleted is leaf:

Simply remove from the tree.



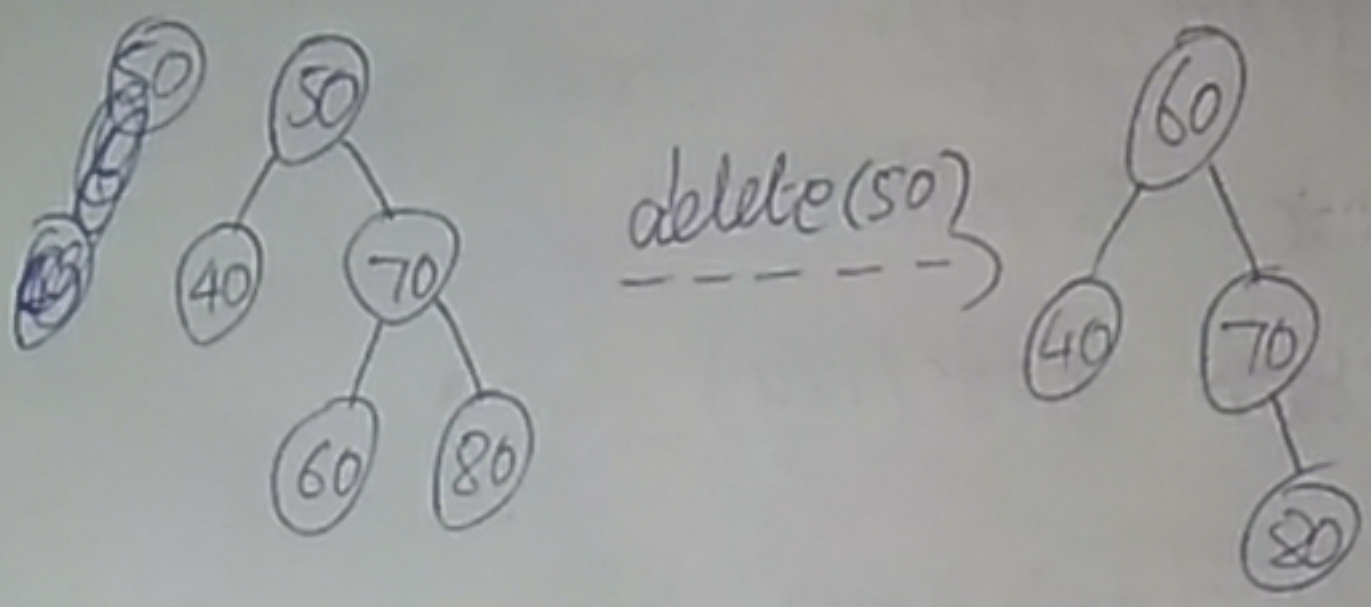
Node to be deleted has only one child:

Copy the child to the node and delete the child.



Node to be deleted has two children:

find inorder successor of the node. Copy contents of the inorder successor to the node and delete inorder successor. Note that inorder predecessor can also be used.



The important thing to note is, inorder successor is needed only when right child is not empty. In this particular case, inorder successor can be obtained by finding the minimum value in right child of the node.

Implementation:

```
Class Node:

    def __init__(self, key):
        self.key = key
        self.left = None
        self.right = None
```

```
def inorder(root):
    if root is not None:
        inorder(root.left)
        print(root.key)
        inorder(root.right)

def insert(node, key):
    if node is None:
        return Node(key)
    if key < node.key:
        node.left = insert(node.left, key)
    else:
        node.right = insert(node.right, key)
    return node

def deletenode(root, key):
    if root is None:
        return root
    if key < root.key:
        root.left = deletenode(root.left, key)
    elif (key > root.key):
        root.right = deletenode(root.right, key)
```

```
else:
    if root.left is None:
        temp = root.right
        root = None
        return temp
    elif root.right is None:
        temp = root.left
        root = None
        return temp

    temp = minvaluenode(root.right)
    root.key = temp.key
    root.right = deletenode(root.right, temp.key)
return root.

def minvalue node(node):
    current = node
    while(current.left is not None):
        current = current.left
    return current
```

Input :-

```
root = None
root = insert(root, 50)
root = insert(root, 30)
root = insert(root, 20)
root = insert(root, 40)
root = insert(root, 40)
root = insert(root, 70)
root = insert(root, 60)
root = insert(root, 80)
inorder(root)
print("\n Delete 20")
 root = deletenode(root, 20)
inorder(root)
print("\n Delete 30")
root = deletenode(root, 30)
inorder(root)
print("\n Delete 50")
root = deletenode(root, 50)
inorder(root)
```

Output :

```
20 30 40 50 60 70 80
Delete 20
30 40 50 60 70 80
Delete 30
40 50 60 70 80
```