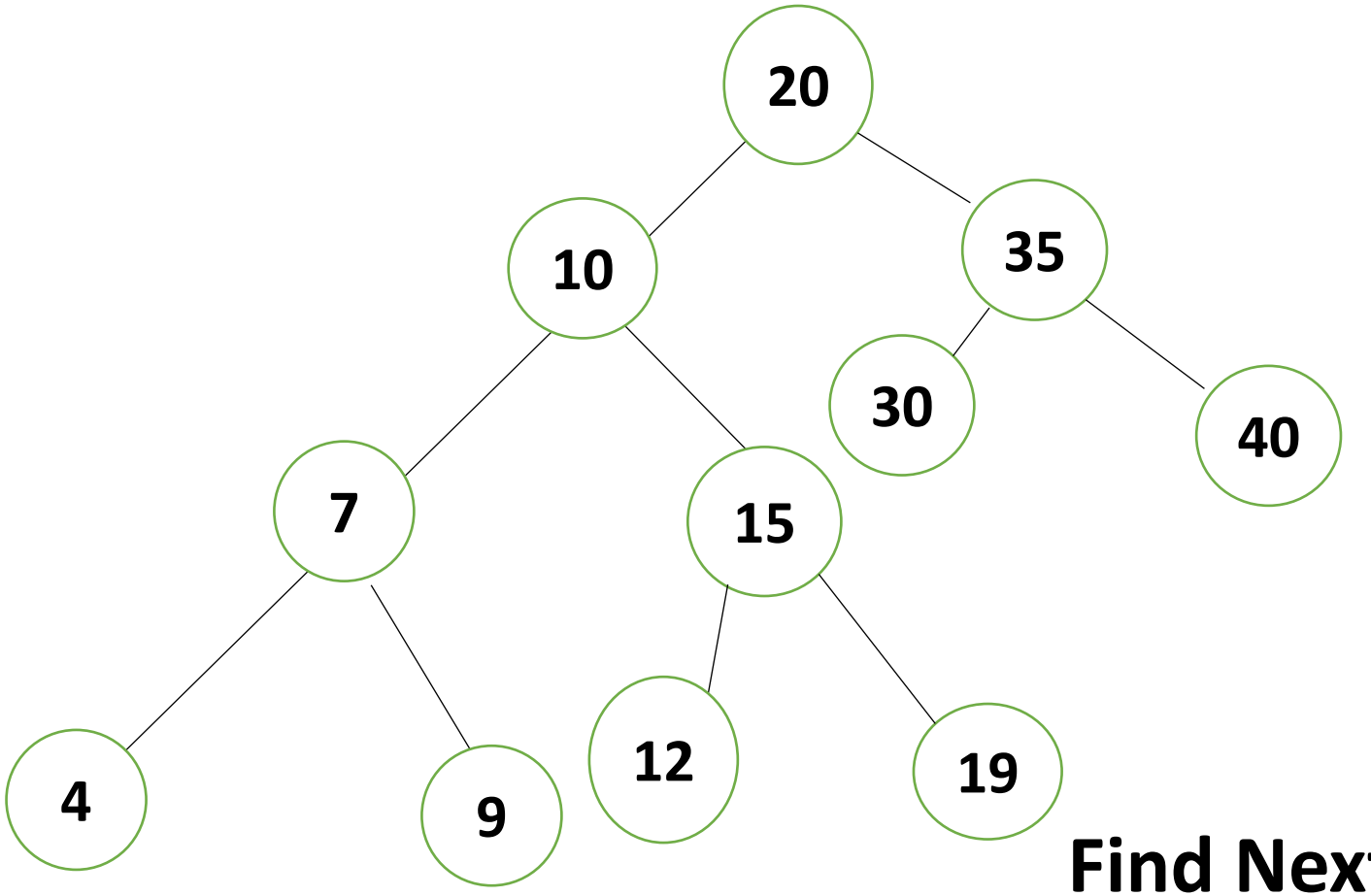
BST: Find Next Node Strategy



Find "Successor" node (if exists):

Given a Node on the tree:
How do we find the "next node"?

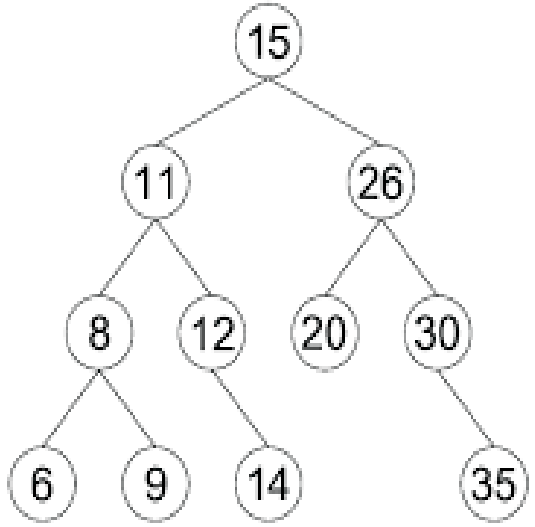
Starting at:
Node 10?
Node 12?
Node 19?

Find Next: Tree Walking Rules:

Next Node Starting at:
(10)? **If RHC exists, Go RHC, walk down all LHC's to Leaf** (12)
(12)? **Else If LHC of Parent, return Parent (15)**
(19)? **Else, Walk-up Parents till Parent X LHC of GParent Y. Use Y** (20)

"Get Next Node" Code

Finding "next" node (if exists): Not-recursive)

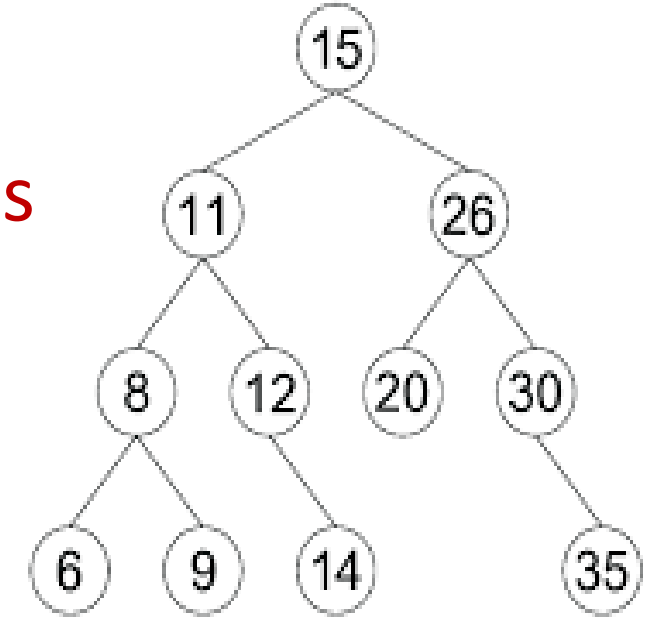



```
Node*BST:: getNext (Node* from) { // Returns next node
    Node *currNode = from→parent;
    if (from->RHC) {   // If RHC, find leftmost leaf (15→20)
        currNode = from→RHC;
        while (currNode→LHC) { currNode = LHC ; }
        return (currNode);   }
 //No RHC.  Is this node the Root.  If so, return error (end of line)
     If (from == toRoot) return (NULL);  // Root and no right hand child.  Done.
// No RHC and not Root?  Is this Node an LHC of Parent?  If so, return Parent
    if (from == from→Parent→LHC))
        return (from→Parent);
 // No RHC, not Root, not an LHC of Parent.
 // Climb tree until parent is LHC of Gparent. Return Gparent (14→15).  Note:  Currnode is from's parent
     while (currnode == currnode→Parent→RHC) {  // Keep climbing
        if (!(currNode = currnode→Parent))  // Keep climbing.  If new parent is Root, we are done
            break;  }
     return (currNode→Parent); }   // Return first Gparent with LHS or Root (where all descent was RHC)
```

Homework 3: Get Previous Node

Deliverables:

1. **List of** "Find Previous Node" Tree Walking rules
2. **Code for**: Node*BST:: getPrevious (Node* from)



Finding the "Preceding" node (if exists):

Node* getPrevious (Node* from) { // Returns previous node or Null (if lowest value)
// Null should be returned only for Node 6 in example

Node 15 → 14
Node 20 → 15