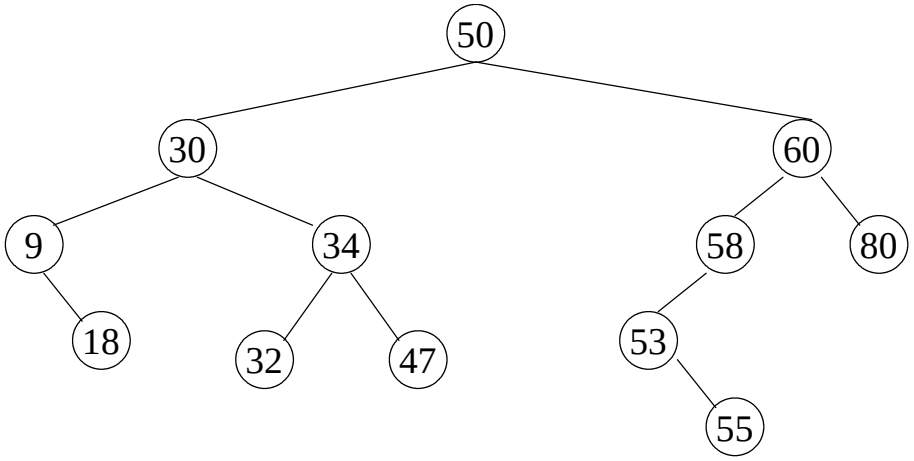
3/12/08 **Lab 8 BST** Name:_____

Objective: To understand how the binary search tree (BST) delete_key method works.

Activity 1: Recall the Binary Search Tree (BST) from yesterday's lecture:



a. What would need to be done to delete 32 from the BST?

b. What would need to be done to delete 9 from the BST?

c. Copy the folder P:\810-063-CSIII\LABS\LAB_8 and open the BST.py file in IDLE. This file contains both the BinarySearchTree and TreeNode classes. Recall that the majority of the work for the delete_key method is done by the TreeNode's delete_key method. In IDLE, run this code to load these class definitions. Build a small BST by executing the following commands.

```
t = BinarySearchTree()
len(t)
t.put(10, 'ten')
len(t)
t.get(10)
t.put(20, 'twenty')
t.put(30, 'thirty')
len(t)
```

Draw the resulting BST.

d. Now delete the node with 20 as the key by "`t.delete_key(20)`". Determine the section of code in the delete_key method that actually deletes the node containing 20.

e. Now delete the node with 10 as the key. What did you observe? (Explain)

f. How might you fix this problem?

After you have answered the above questions, raise your hand and explain your answers.

Activity 2: Fix the bug in the TreeNode class discovered in Activity 1.

After you have fixed this bug, raise your hand and demonstrate your code.

Activity 3: To the TreeNode's `delete_key`, `findSuccessor`, `findMin`, and `spliceOut` methods add print statements to help trace what nodes are being visited by each method when a delete_key is performed. Open and run the file buildBST.py that:

- constructs the BST on page 1,
- deletes the key 9
- deletes the key 32
- deletes the key 50
- checks to see if the remaining key still exists

When your print statements thoroughly demonstrate that the code is working correctly, raise your hand and demonstrate buildBST.py.

NOTE: If you complete all of the activities within the lab period, you do not need to hand anything it. However, if you need to finish activities outside of the lab period, then hand in written questions to activities not completed during the lab period.

```
    def findSuccessor(self):
        succ = None
        if self.rightChild:
            succ = self.rightChild.findMin()
        else:
            if self.parent.leftChild == self:
                succ = self.parent
            else:
                self.parent.rightChild = None
                succ = self.parent.findSuccessor()
                self.parent.rightChild = self
        return succ

    def findMin(self):
        n = self
        while n.leftChild:
            n = n.leftChild
        print 'found min, key = ', n.key
        return n

    def spliceOut(self):
        if (not self.leftChild and not self.rightChild):
            if self == self.parent.leftChild:
                self.parent.leftChild = None
            else:
                self.parent.rightchild = None
        elif (self.leftChild or self.rightChild):
            if self.leftChild:
                if self == self.parent.leftChild:
                    self.parent.leftChild = self.leftChild
                else:
                    self.parent.rightChild = self.leftChild
            else:
                if self == self.parent.leftChild:
                    self.parent.leftChild = self.rightChild
                else:
                    self.parent.rightChild = self.rightChild
```

```
    def delete_key(self,key):
         if self.key == key: # do the removal
             if not (self.leftChild or self.rightChild):
                 if self == self.parent.leftChild:
                     self.parent.leftChild = None
                 else:
                     self.parent.rightChild = None
             elif (self.leftChild or self.rightChild) and \
                      (not (self.leftChild and self.rightChild)):
                 if self.leftChild:
                     if self == self.parent.leftChild:
                         self.parent.leftChild = self.leftChild
                     else:
                         self.parent.rightChild = self.leftChild
                 else:
                     if self == self.parent.leftChild:
                         self.parent.leftChild = self.rightChild
                     else:
                         self.parent.rightChild = self.rightChild
             else:  # replace self with successor
                 succ = self.findSuccessor()
                 succ.spliceOut()
                 if self == self.parent.leftChild:
                     self.parent.leftChild = succ
                 else:
                     self.parent.rightChild = succ
                 succ.leftChild = self.leftChild
                 succ.rightChild = self.rightChild
         else: # continue looking
             if key < self.key:
                 if self.leftChild:
                     self.leftChild.delete_key(key)
                 else:
                     print "error"
             else:
                 if self.rightChild:
                         self.rightChild.delete_key(key)
                 else:
                         print "trying to remove a non-existant node"
```