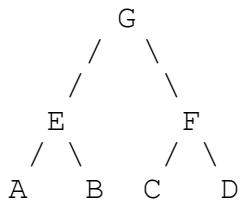
Unit 9 Practice Test

____ 1. If the tree below is listed by postorder traversal, what will the result be?



A. G E F A B C D
B. G E A B F C D
C. A E B G C F D
D. A E B C F D G
E. A B E C D F G

____ 2. The following array represents a max-at-top heap:

34 25 18 16 12 5 2

Two remove operations are performed on this heap, and the value 20 is then inserted. What does the array representation of the heap look like at the end of these operations?

A. 34 25 18 16 12 20
B. 34 25 20 18 16 12
C. 34 25 20 16 12 18
D. 20 16 18 2 12 5
E. 20 18 16 12 5 2

____ 3. Given the hash table shown at right, items with key values of "left" and "under" are inserted in that order using *quadratic* probing and a hash function that assigns a hash value equal to the number of characters in the key. The new items will end up in which positions of the table (if any)?

A. "left" in position 0, and "under" in position 1
B. "left" in position 1, and "under" in position 3
C. "left" in position 1, and "under" in position 0
D. "left" in position 1, but overflow occurs for "under"
E. overflow occurs for both "left" and "under"

0	
1	
2	up
3	
4	down
5	right
6	behind

Questions 4 and 5 refer to binary trees whose nodes have the following structure:

```
public class Node {
    public char val;
    public Node left;
    public Node right;
}
```

____ 4. What does the following function do to the tree whose root is referred to by the parameter `root`?

```
public static int mystery(Node root) {
     int sum;
     if (root != null) {
        if (root.left == null && root.right == null)
            return 0;
        sum = mystery(root.left) + mystery(root.right);
        if (sum == 0) {
            Node temp = root.left;
            root.left = root.right;
            root.right = temp;
        }
     }
     return -1;
}
```

A. swaps the left and right subtrees of the root node
B. swaps any two leaf nodes that have the same parent
C. swaps the leftmost and rightmost leaves
D. replaces the tree with its mirror image
E. leaves the tree unchanged

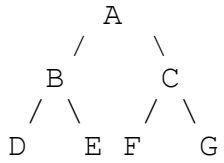
____ 5. Consider the following functions:

```
public void preorder(Node root, Stack s) {
    if (root != null) {
        s.push(root.val);
        preorder(root.left, s);
        preorder(root.right, s);
    }
}

public void postorder(Node root, Stack s)  {
    if (root != null) {
        postorder(root.left, s);
        postorder(root.right, s);
        root.val = s.pop();
    }
}
```

where `push` and `pop` are methods of a class that implements our `Stack` interface and that represents a stack of characters. `push(val)` pushes the character stored in `val` onto the top of the stack, and `pop()` pops the top character in the stack and returns it.

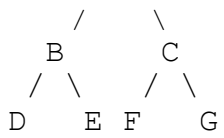
If `root` initially points to the following tree



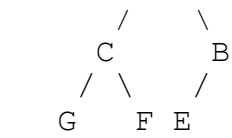
what tree results from the execution of the following statements?

```
Stack s = new Stack();
preorder(root, s);
postorder(root, s);
```

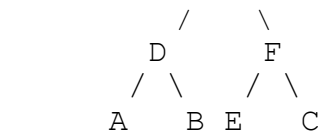
A.



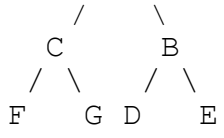
B.



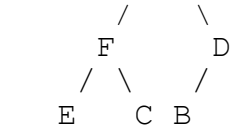
C.



D.



E.

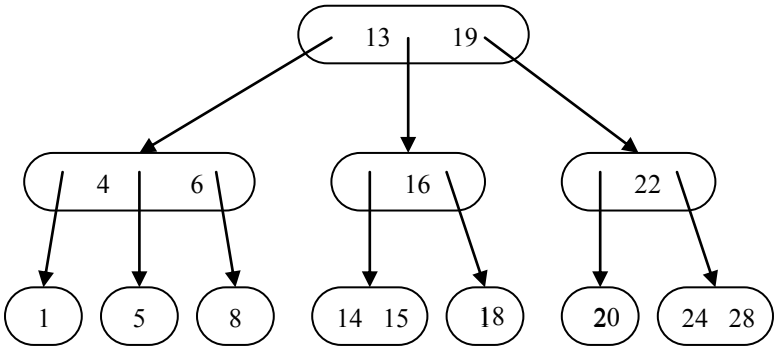


6. The following array is to be sorted using heap sort.

```
6   17   22    9    4    16    12
```

Show the contents of the array after the initial, heap-creation phase of the algorithm. Make sure to give your final answer as an array.

7. The following is a 2-3 tree:



Draw what the tree will look like after performing the following sequence of operations:

insert 10
insert 30
insert 27
insert 25

8. Write a Java method called `countOccurrences()` that counts the number of times that a specified character appears in a binary tree that has been built using instances of the `Node` class from questions 4 and 5. Your method should be static, and it should take two arguments: (1) a reference to the root node of the tree, and (2) the value of type `char` for which you are searching. You should *not* assume that the tree is binary *search* tree.