AP Computer Science
Practice Test 8 - Linked Lists and Binary Search

Name _____
Date _____ Period _____

1) Create a binary search tree using the list of **Strings** below. Use the resulting binary search tree to identity the root of the tree. Then identify a **subtree**, a single **parent** and **child** pair, and a **leaf**.

Oh Boy One More Tree For Me Gee Please Set Us Free Is Our Plea

2) Suppose in a non-empty binary tree each node is either a leaf or has two children. Which statement is true?

I. The total number of nodes is odd.
II. The number of nodes is $2^k - 1$ for some positive integer k.
III. The number of leaves is one more than the number of non-leaves.

A. I only
B. II only
C. I and III only
D. II and III only
E. I, II, and III

3) Consider the following method for removing a value (**anode**) from a linked list:

```
public void remove (ListNode<Integer> anode)
{
        ListNode<Integer> temp = anode.getNext();
        anode.setValue(temp.getValue());
        anode.setNext(temp.getNext());
}
```

In which of the following cases will the **remove** method fail to work as intended?

I. **anode** points to any node in the list other than the first or last node.
II. **anode** points to the last node in the list.
III. **anode** points to the first node, and there is more than one node in the list.

(A) I only (C) I and II only (E) I, II, and III

(B) II only (D) I and III only

4) The following method is supposed to search for and remove from a linked list all nodes whose data fields are equal to **`value`**, a previously defined value. Assume that **`first`** is accessible and references the first node in the list.

```
public void removeNodesWithValue(String value)
{
      ListNode<String> previous = null;
      ListNode<String> current = first;
      while (current != null)
      {
            if (current.getValue().equals(value))
            {
                  previous.setNext(current.getNext());
            }
            else
            {
                  previous = current;
            }
            current = current.getNext();
      }
}
```

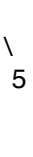
What is true about the above method?

A. It works for all the nodes of the linked list.
B. It fails for only the first node of the list (if the value is found at ListNode **first**).
C. It fails for only the last node of the list.
D. It fails for the first and last nodes of the list, but works for all others.
E. It fails for all nodes in the list.

5) Consider the following method:

```
public String guessNumber(TreeNode<Integer> node)
{
      if (node != null)
            return node.getValue() + guessNumber(node.getLeft())
                  + guessNumber(node.getRight());
      return “”;
}
```

If **`root`** points to the root of the following tree:



what string will be returned by `guessNumber(root)`?

A. 12345
B. 42135
C. 13254
D. 42513
E. 54321

6) Write a **recursive** method `reverseList()` that reverses the pointers in the list. (For example, a list 1 -> 2 -> 3 -> 4 -> 5 -> 6 would be reversed 6 -> 5 -> 4 -> 3 -> 2 -> 1). It is called from the main method as `reverseList(null, first)`. Be sure to change the list pointers `first` and `last`.

```
public void reverseList(ListNode<Integer> prior, ListNode<Integer> theNode)
```

7) Consider the following code fragment.

```
ListNode<Integer> first, p, t;
t = new ListNode<Integer>(new Integer(150));
first = t;
p = new ListNode<Integer>(new Integer(100));
t.setNext(p);
t = p;
p = new ListNode<Integer>(new Integer(120));
p.setNext(first);
first = p;
p = new ListNode<Integer>(new Integer(140));
t.setNext(p);
t = p;
p = new ListNode<Integer>(new Integer(130));
t.setNext(p);
```

What would be printed by the following code fragment?

```
ListNode<Integer> node = first;
while (node != null)
{
        System.out.print(node.getValue() + " ");
        node = node.getNext();
}
```

(A) 150 100 120 140 130

(B) 150 120 100 140 130

(C) 120 150 100 140 130

(D) 130 140 120 150 100

(E) NullPointerException

8) Write a method **`public ListNode<E> removeLast ( )`** that removes the last (non-null) node in the list. In doing this, the second-to-last node should become the last node. Return this (new) last node. Assume that `ListNode first` and `ListNode last` are data members in the `SinglyLinkedlist` that contains this method. Your method should work even if there is only one valid (non-null) node in the list, or if the list is empty.

```
public ListNode<E> removeLast ( )
```

9) You have a singly-linked list and the list contains the node `inList`. Fill in the blanks so that the method `insertNode()` inserts the node `addMe` **after** the node `inList` in the list.

```
public void insertNode(ListNode<Double> addMe, ListNode<Double> inList)
{
      _____________________.setNext(_____________________.getNext());

      _____________________.setNext(_____________________);
}
```

10) **`head`** points to the first ListNode of a singly-linked list. Which of the following code segments adds a new node with the Integer value 534 as the first node in the list.

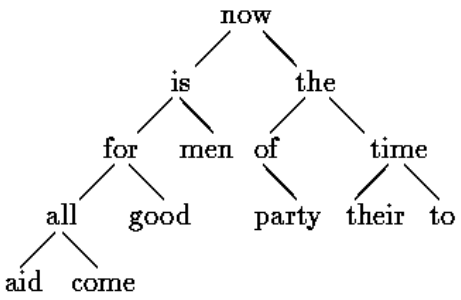
```
I.    ListNode<Integer> oldNode = head;
      head = new ListNode<Integer>(new Integer(534), null);
      head.setNext(oldNode);

II.   Integer num = new Integer(534);
      ListNode<Integer> newNode = new ListNode<Integer>(num, head.getNext());
      head.setNext(newNode);
      head = newNode;

III.  head = new ListNode<Integer>(new Integer(534), head);
```

A. I only
B. II only
C. I and II only
D. I and III only
E. I, II, and III

11) Write method (recursive or non-recursive) `public TreeNode minNode (TreeNode root)` that finds and returns the smallest node (the node with the smallest value) in the Binary Search Tree.

```
public TreeNode<Integer> minNode (TreeNode<Integer> root)
```

12) Write the tree at right inorder and postorder.



INORDER:

POSTORDER:

13) Write the following expression in postfix notation.

57 ^ (20 - 28) / 12 ^ 8 + 91 * (7 - 14) * 84 / 79 - 5

POSTFIX:

14) Draw an expression tree for the expression in problem 13.