Studying
Do many questions on Practice-It! First, write your solution down on paper. Then, debug it by hand. Finally, type it into Practice-It! to see what you got wrong.

Question Types
The following types of questions might appear on the exam

Binary Search Trees
• Given a set of values, add them to a binary search tree.
• Perform traversals in the three standard orders on a tree.

Polymorphism Mystery
Given a set of classes with inheritance relationships, a set of variables declared using those classes, and a set of method calls made on those variables, determine the output.

Comparable Programming
Write a complete class and make it Comparable based on a given set of comparison criteria.

Collections Programming
Write a method that uses one or more class from the Java Collections Framework.

(Easier) Binary Tree Programming
Add a method to the IntTree class from lecture.

(Harder) Binary Tree Programming
Add a method to the IntTree class from lecture

Linked List Programming
Add a method to the LinkedIntList class from lecture.

Recursive Programming
Write a method that uses recursion.

Untested Topics
The following topics will definitely not appear on the exam

2-D arrays Detailed Knowledge of Big-Oh Running/Re-writing Searching and Sorting Algorithms
Recursive Backtracking Catching Exceptions Priority Queues
Huffman Coding IO Streams Abstract/Inner Classes
Hashing Implementing Iterators Implementing a “generic” class
**Linked Lists**

Write a method called `moveSecondToLastToFront` that rearranges the order of a list of integers so that the second to last element of the list appears at the front. For example, if a variable called `list` stores these values:

\[
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
\]

and you make the call `list.moveSecondToLastToFront()`, the list should be the following:

\[
[8, 0, 1, 2, 3, 4, 5, 6, 7, 9]
\]

If the list has fewer than two elements, it should be unchanged by a call to `moveSecondToLastToFront`. You are writing a public method for a linked list class defined as follows:

```java
public class ListNode {
    public int data; // data stored in this node
    public ListNode next; // link to next node in the list
    <constructors>
}
public class LinkedIntList {
    private ListNode front;
    <methods>
}
```

You are writing a method that will become part of the `LinkedIntList` class. You may define private helper methods to solve this problem, but, otherwise, you may not assume that any particular methods are available. You are allowed to define your own variables of type `ListNode`, but you may not construct any new nodes, and you may not use any auxiliary data structure to solve this problem (no array, ArrayList, stack, queue, String, etc). You also may not change any data fields of the nodes. You MUST solve this problem by rearranging the links of the list. Your solution must run in \(O(n)\) time where \(n\) is the length of the list.
Binary Trees

Write a method called makeEvenTree for a binary tree of integers. The method should return the sum of all the integers in the tree augmented in the following way:

- Even numbers should be counted normally
- Odd numbers should be counted twice

For example, if a variable tree stores a reference to the following tree:

```
1
/  \
2   4
/ \   /
3  6  5
```

then the call tree.makeEvenTree() should return $1 + 1 + 2 + 3 + 3 + 6 + 4 + 5 + 5 = 30$.

You are writing a public method for a binary tree class defined as follows:

```java
public class IntTreeNode {
    public int data; // data stored in this node
    public IntTreeNode left; // reference to left subtree
    public IntTreeNode right; // reference to right subtree
    <constructors>
}

public class IntTree {
    private IntTreeNode overallRoot;
    <methods>
}
```

You may define private helper methods to solve this problem, but, otherwise, you may not call any other methods of the class. You may not define any auxiliary data structures to solve this problem.
Binary Trees

Write a method called makeEvenTree for a binary tree of integers. The method should replace all the odd values in the tree with their twice their value. For example, if a variable tree stores a reference to the following tree:

```
1
  2
    3
  4
    5
```

then, after the call tree.makeEvenTree(), tree should store a reference to the following tree:

```
2
  2
    6
  4
    10
```

You are writing a public method for a binary tree class defined as follows:

```java
public class IntTreeNode {
    public int data; // data stored in this node
    public IntTreeNode left; // reference to left subtree
    public IntTreeNode right; // reference to right subtree
    <constructors>
}

public class IntTree {
    private IntTreeNode overallRoot;
    <methods>
}
```

You may define private helper methods to solve this problem, but, otherwise, you may not call any other methods of the class. You may not define any auxiliary data structures to solve this problem.