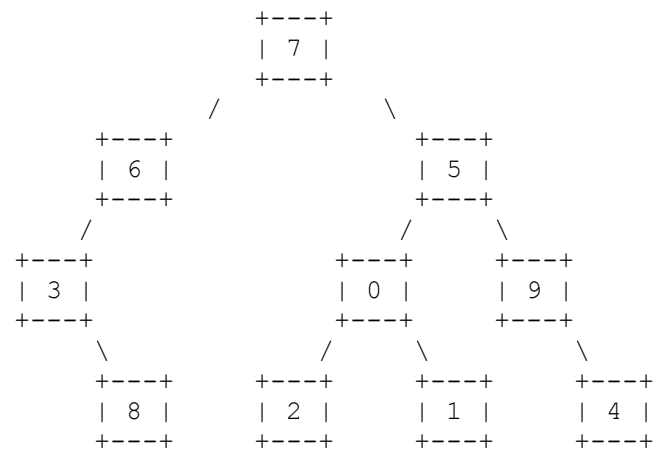
CSE143 Section #17 Problems

1. Binary Tree Traversals, 6 points. Consider the following tree.



Fill in each of the traversals below:

Preorder traversal _____

Inorder traversal _____

Postorder traversal _____

2. Binary Search Tree, 4 points. Draw a picture below of the binary search tree that would result from inserting the following words into an empty binary search tree in the following order: Java, Rust, Kotlin, Swift, Python, C, Go. Assume the search tree uses alphabetical ordering to compare words.

3. Collections Programming, 15 points. Write a method called recordDate that records information about a date between two people. For each person, the map records an ordered list of people that person has dated. For example, the map might record these entries for two people

```
Michael => [Ashley, Samantha, Joshua, Brittany, Amanda, Amanda]
Amanda  => [Michael, Daniel, Michael]
```

The dates are listed in reverse order. The list for Michael indicates that he most recently dated Ashley and before that Samantha and before that Joshua, and so on. Notice that he has dated Amanda twice. The list for Amanda indicates that she most recently dated Michael and before that Daniel and before that Michael. All names are stored as string values.

The method takes three parameters: the map, the name of the first person, and the name of the second person. It should record the date for each person and should return what date number this is (1 for a first date, 2 for a second date, and so on). Given the entries above, if we make this call:

```
int n = recordDate(dates, "Michael", "Amanda");
```

The method would record the new date at the front of each list:

```
Michael => [Amanda, Ashley, Samantha, Joshua, Brittany, Amanda, Amanda]
Amanda  => [Michael, Michael, Daniel, Michael]
```

The method would return the value 3 indicating that this is the third date for this pair of people. When someone is first added to the map, you should construct a LinkedList object (we use LinkedList instead of ArrayList because it has fast insertion at the front of the list).

4. Comparable class, 15 points. Define a class called FoodData that stores information about a food item. Each FoodData object keeps track of the name of the food along with its number of grams of fat, carbohydrate, and protein. For example:

```
FoodData item1 = new FoodData("sausage biscuit", 31.0, 39.0, 11.0);
FoodData item2 = new FoodData("strawberry sundae", 6.0, 49.0, 6.0);
FoodData item3 = new FoodData("banana", 0.4, 31.1, 1.5);
```

In calling the constructor, the name is followed by fat grams, followed by carbohydrate grams, followed by protein grams. For example, the sausage biscuit above has 31 grams of fat, 39 grams of carbohydrate, and 11 grams of protein. As in the third example, these values can be real numbers. Your constructor should throw an IllegalArgumentException if any of the numeric values passed to it is negative.

This class is being designed for programs that will help people who want to use a low-fat diet. For example, it is a bit surprising that the McDonalds sausage biscuit (item1) gets over 58% of its calories from fat while the McDonalds strawberry sundae (item2) gets only around 19.7% of its calories from fat. The banana (item3) gets less than 2.7% of its calories from fat.

Your class should have the following public methods:

getName()	returns the name of this food item
getCalories()	returns total calories for this food item
percentFat()	returns the percent of calories from fat for this item
toString()	returns a String representation of this item

To compute total calories and percent fat, assume that each gram of fat is 9 calories and that each gram of carbohydrate and protein is 4 calories. For example, the calories for the strawberry sundae (item2 above) are 274:

```
9 * (fat grams) + 4 * (carb grams) + 4 * (protein grams) =
9 * 6.0 + 4 * 49.0 + 4 * 6.0 = 274.0
```

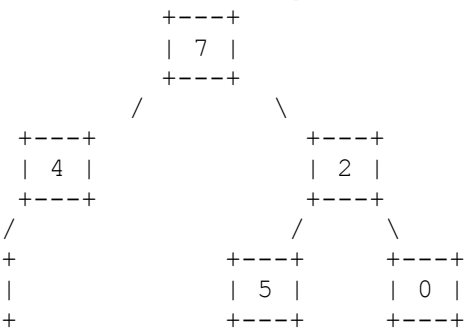
Its percent fat is a little over 19.7 because the calories from fat are 54 and the total calories are 274. As usual, percentages should be expressed as real numbers in the range of 0.0 to 100.0. You may assume that the number of calories will be greater than 0.

The toString method should include the name of the item followed by a colon followed by the fat, carbohydrate and protein grams, separated by commas, and labeled, as in the following examples for item1, item2, and item3 above:

```
sausage biscuit: 31.0g fat, 39.0g carbohydrates, 11.0g protein
strawberry sundae: 6.0g fat, 49.0g carbohydrates, 6.0g protein
banana: 0.4g fat, 31.1g carbohydrates, 1.5g protein
```

Your class should also implement the Comparable interface. Items should be ordered first by percent of calories coming from fat (lowest to highest) and then by name (sorted alphabetically).

5. Binary Trees, 10 points. Write a method of the IntTree class called inorderList that returns a list containing the sequence of values obtained from an inorder traversal of the tree. For example, if a variable t stores a reference to the following tree:



then the call t.inorderList() should return the following list:
[9, 4, 7, 5, 2, 0]
Your method should construct an ArrayList to return. If the tree is empty, your method should return an empty list.

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

```
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 are writing a method that will become part of the IntTree class. 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 construct any extra data structures other than the ArrayList you are returning.

6. Details of inheritance, 10 points. Assuming that the following classes have been defined:

```
public class Cup extends Box {
    public void method1() {
        System.out.println("Cup 1");
    }

    public void method2() {
        System.out.println("Cup 2");
        super.method2();
    }
}

public class Pill {
    public void method2() {
        System.out.println("Pill 2");
    }
}

public class Jar extends Box {
    public void method1() {
        System.out.println("Jar 1");
    }

    public void method2() {
        System.out.println("Jar 2");
    }
}

public class Box extends Pill {
    public void method2() {
        System.out.println("Box 2");
    }

    public void method3() {
        method2();
        System.out.println("Box 3");
    }
}
```

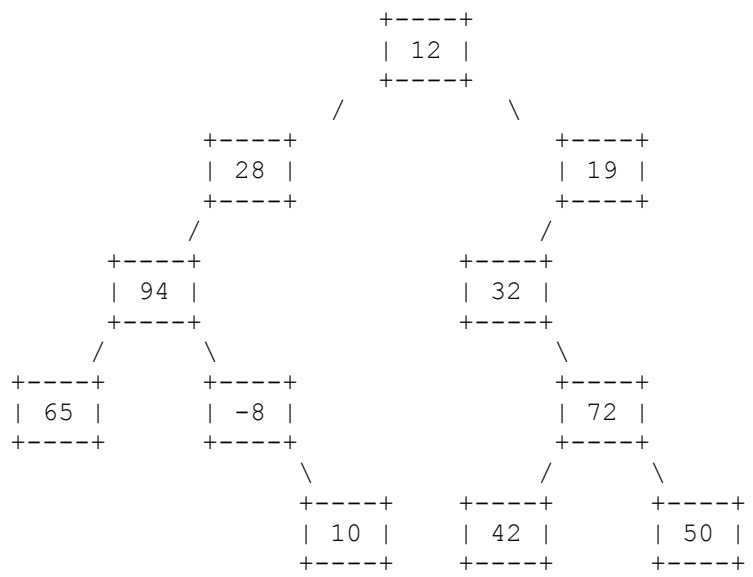
And assuming the following variables have been defined:

```
Box var1 = new Box();
Pill var2 = new Jar();
Box var3 = new Cup();
Box var4 = new Jar();
Object var5 = new Box();
Pill var6 = new Pill();
```

In the table below, indicate in the right-hand column the output produced by the statement in the left-hand column. If the statement produces more than one line of output, indicate the line breaks with slashes as in "a/b/c" to indicate three lines of output with "a" followed by "b" followed by "c". If the statement causes an error, fill in the right-hand column with either the phrase "compiler error" or "runtime error" to indicate when the error would be detected.

Statement	Output
`var1.method2();`	_____
`var2.method2();`	_____
`var3.method2();`	_____
`var4.method2();`	_____
`var5.method2();`	_____
`var6.method2();`	_____
`var1.method3();`	_____
`var2.method3();`	_____
`var3.method3();`	_____
`var4.method3();`	_____
`((Cup)var1).method1();`	_____
`((Jar)var2).method1();`	_____
`((Cup)var3).method1();`	_____
`((Cup)var4).method1();`	_____
`((Jar)var4).method2();`	_____
`((Box)var5).method2();`	_____
`((Pill)var5).method3();`	_____
`((Jar)var2).method3();`	_____
`((Cup)var3).method3();`	_____
`((Cup)var5).method3();`	_____

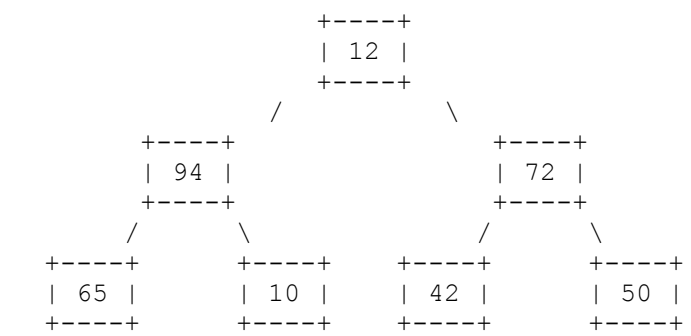
7. Binary Trees, 20 points. Write a method of the IntTree class called tighten that eliminates branch nodes that have only one child from the tree. For example, if a variable called t stores a reference to the following tree:



then the call:

```
t.tighten();
```

should leave t storing the following tree:



Notice that the nodes that stored the values 28, 19, 32, and -8 have all been eliminated from the tree because each had one child. When a node is removed, it is replaced by its child. Notice that this can lead to multiple replacements because the child might itself be replaced (as in the case of 19 which is replaced by its child 32 which is replaced by its child 72).

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

```
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 are writing a method that will become part of the IntTree 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 not allowed to change the data fields of the existing nodes in the tree (what we called "morphing" in assignments 7 and 8), you are not allowed to construct new nodes or additional data structures, and your solution must run in O(n) time where n is the number of nodes in the tree.

8. Linked Lists, 20 points. Write a method of the LinkedIntList class called shiftLastOf3 that shifts the third value in each successive group of three from a list of integers to the front of the list, preserving the relative order of the values and returning a count of the number of nodes that were shifted. For example, if a variable called list stores these values:

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
 |     |  |     |  |     |  |      |
 +-----+  +-----+  +-----+  +------+
  group    group    group    group
```

Then after the call:

```
int count = list.shiftLastOf3();
```

the list should store the following values:

```
[2, 5, 8, 11, 0, 1, 3, 4, 6, 7, 9, 10]
```

and the variable count would be set to 4. Notice that the original list has four groups of three values. The last value in each group (2, 5, 8, and 11) has been shifted to the front of the list but otherwise all values still have the same relative ordering as they did in the original list. Because four values were shifted, the method returns a value of 4.

This example uses consecutive integers to make it easier to see the effect of the shifting, but you should not make any assumptions about the values in the list. It also has no stray values at the end. If there are extra values at the end of a list that don't make a complete group of three, then they are not shifted. For example, if a list contains fewer than three values, then no values are shifted and the method would return a count of 0.

As another example, if list had instead stored these values:

```
[3, 19, 7, 45, -2, 8, 6, 18, 42, 5, 12]
 |      |   |      |  |       |
 +------+   +------+  +-------+
  group      group      group
```

then after the method is called, it would store these values:

```
[7, 8, 42, 3, 19, 45, -2, 6, 18, 5, 12]
```

and the method would return 3 to indicate that three values were shifted.

You are writing a public method for a linked list class defined as follows:

```
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, you may not use any auxiliary data structure to solve this problem (no array, ArrayList, stack, queue, String, etc), and your solution must run in O(n) time where n is the number of nodes in the list. You also may not change any data fields of the nodes. You MUST solve this problem by rearranging the links of the list.