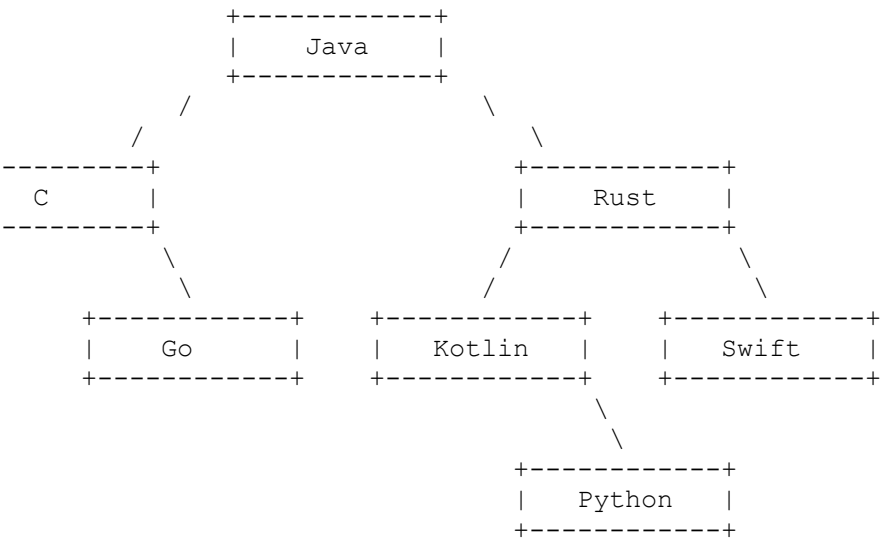
CSE143 Section #17 Solutions

1. Preorder traversal 7, 6, 3, 8, 5, 0, 2, 1, 9, 4
   Inorder traversal 3, 8, 6, 7, 2, 0, 1, 5, 9, 4
   Postorder traversal 8, 3, 6, 2, 1, 0, 4, 9, 5, 7

2. 

3. One possible solution appears below.

```
public static int recordDate(Map<String, List<String>> dates,
                             String name1, String name2) {
    if (!dates.containsKey(name1)) {
        dates.put(name1, new LinkedList<String>());
    }
    if (!dates.containsKey(name2)) {
        dates.put(name2, new LinkedList<String>());
    }
    dates.get(name1).add(0, name2);
    dates.get(name2).add(0, name1);
    int n = 0;
    for (String s : dates.get(name1)) {
        if (s.equals(name2)) {
            n++;
        }
    }
    return n;
}
```

4. One possible solution appears below.

```
public class FoodData implements Comparable<FoodData> {
    private String name;
    private double fat;
    private double carbs;
    private double protein;

    public FoodData(String name, double fat, double carbs,
                    double protein) {
        if (fat < 0 || carbs < 0 || protein < 0) {
            throw new IllegalArgumentException();
        }
        this.name = name;
        this.fat = fat;
        this.carbs = carbs;
        this.protein = protein;
    }

    public String getName() {
        return name;
    }

    public double getCalories() {
        return 9 * fat + 4 * (carbs + protein);
    }

    public double percentFat() {
        return 100.0 * (9 * fat / getCalories());
    }

    public String toString() {
        return name + ": " + fat + "g fat, " + carbs +
               "g carbohydrates, " + protein + "g protein";
    }

    public int compareTo(FoodData other) {
        double difference = percentFat() - other.percentFat();
        if (difference < 0) {
            return -1;
        } else if (difference > 0) {
            return 1;
        } else {
            return name.compareTo(other.name);
        }
    }
}
```

5. One possible solution appears below.

```
public List<Integer> inorderList() {
    List<Integer> result = new ArrayList<Integer>();
    inorderList(result, overallRoot);
    return result;
}

private void inorderList(List<Integer> result, IntTreeNode root) {
    if (root != null) {
        inorderList(result, root.left);
        result.add(root.data);
        inorderList(result, root.right);
    }
}
```

6.

Statement	Output
var1.method2()	Box 2
var2.method2()	Jar 2
var3.method2()	Cup 2/Box 2
var4.method2()	Jar 2
var5.method2()	compiler error
var6.method2()	Pill 2
var1.method3()	Box 2/Box 3
var2.method3()	compiler error
var3.method3()	Cup 2/Box 2/Box 3
var4.method3()	Jar 2/Box 3
((Cup)var1).method1()	runtime error
((Jar)var2).method1()	Jar 1
((Cup)var3).method1()	Cup 1
((Cup)var4).method1()	runtime error
((Jar)var4).method2()	Jar 2
((Box)var5).method2()	Box 2
((Pill)var5).method3()	compiler error
((Jar)var2).method3()	Jar 2/Box 3
((Cup)var3).method3()	Cup 2/Box 2/Box 3
((Cup)var5).method3()	runtime error

7. One possible solution appears below.

```
public void tighten() {
    overallRoot = tighten(overallRoot);
}

private IntTreeNode tighten(IntTreeNode root) {
    if (root != null) {
        root.left = tighten(root.left);
        root.right = tighten(root.right);
        if (root.left == null && root.right != null) {
            root = root.right;
        } else if (root.left != null && root.right == null) {
            root = root.left;
        }
    }
    return root;
}
```

8. One possible solution appears below.

```
    public int shiftLastOf3() {
        int count = 0;
        if (front != null && front.next != null && front.next.next != null) {
            ListNode curr1 = front;
            ListNode oldFront = curr1;
            front = front.next.next;
            ListNode curr2 = front;
            curr1.next.next = curr2.next;
            curr1 = curr1.next.next;
            count++;
            while (curr1 != null && curr1.next != null &&
                   curr1.next.next != null) {
                curr2.next = curr1.next.next;
                curr2 = curr2.next;
                curr1.next.next = curr2.next;
                curr1 = curr1.next.next;
                count++;
            }
            curr2.next = oldFront;
        }
        return count;
    }
```