CSE 143 Final Solutions
Summer 2019

```
1.     preorder:  8 7 2 4 9 5 1 3 6 0
       inorder:   4 2 9 7 8 3 1 6 5 0
       postorder: 4 9 2 7 3 6 1 0 5 8
```

```
2.                                 +-------------+
                                   |    Jason    |
                                   +-------------+
                                  /                \
                                 /                  \
                 +-------------+                    +-------------+
                 |    Frank    |                    |     Leo     |
                 +-------------+                    +-------------+
                /                \                                  \
               /                  \                                  \
+-------------+                   +-------------+                     +-------------+
|  Annabeth   |                   |    Hazel    |                     |    Piper    |
+-------------+                   +-------------+                     +-------------+
                                                                     /
                                                                    /
                                                    +-------------+
                                                    |    Percy    |
                                                    +-------------+
```

3. One possible solution appears below

```
    public static double recordScore(Map<String, List<Integer>> scores,
                                     String name, int score) {
        if (score < 0 || score > 20) {
            throw new IllegalArgumentException();
        }
        if (!scores.containsKey(name)) {
            scores.put(name, new LinkedList<Integer>());
        }
        scores.get(name).add(0, score);
        int total = 0;
        for (int i : scores.get(name)) {
            total += i;
        }
        return (100.0 * total / (scores.get(name).size() * 20.0));
    }
```

4. One possible solution appears below.

```
public class BookData implements Comparable<BookData> {
    private String title;
    private String author;
    private int reviews;
    private double total;

    public BookData(String title, String author) {
        this.title = title;
        this.author = author;
        this.reviews = 0;
        this.total = 0.0;
    }

    public void review(double rating) {
        reviews++;
        total += rating;
    }

    public String getTitle() {
        return title;
    }

    public double getRating() {
        if (reviews == 0)
            return 0.0;
        else
            return total / reviews;
    }

    public String toString() {
        double rating = (int) (10.0 * getRating()) / 10.0;
        String result = title + ", by " + author + ", " + rating + " (";
        if (reviews == 1)
            result += "1 review)";
        else
            result += reviews + " reviews)";
        return result;
    }

    public int compareTo(BookData other) {
        double delta = other.getRating() - getRating();
        if (delta < 0)
            return -1;
        else if (delta > 0)
            return 1;
        else // delta == 0
            return other.reviews - reviews;
    }
}
```

5. One possible solution appears below.

```
    public void printLeaves() {
        if (overallRoot == null) {
            System.out.println("no leaves");
        } else {
            System.out.print("leaves:");
            printLeaves(overallRoot);
            System.out.println();
        }
    }

    private void printLeaves(IntTreeNode root) {
        if (root != null) {
            if (root.left == null && root.right == null) {
                System.out.print(" " + root.data);
            } else {
                printLeaves(root.right);
                printLeaves(root.left);
            }
        }
    }
```

6.

```
        Statement                       Output
        ------------------------------------------------------------
        var1.method1();                 compiler error
        var2.method1();                 Poptart 1
        var1.method2();                 Hotdog 2/Sandwich 2/Hotdog 3
        var2.method2();                 Poptart 2
        var3.method2();                 Hotdog 2/Sandwich 2/SeattleDog 3
        var4.method2();                 compiler error
        var5.method2();                 Hotdog 2/Sandwich 2/SeattleDog 3
        var1.method3();                 Hotdog 3
        var2.method3();                 Sandwich 3
        var3.method3();                 SeattleDog 3
        var4.method3();                 compiler error
        var5.method3();                 SeattleDog 3
        ((Poptart)var3).method1();      runtime error
        ((SeattleDog)var5).method1();   SeattleDog 1
        ((Hotdog)var3).method1();       compiler error
        ((Hotdog)var3).method3();       SeattleDog 3
        ((SeattleDog)var6).method3();   runtime error
        ((Sandwich)var4).method2();     Sandwich 2
        ((Hotdog)var4).method3();       runtime error
        ((Poptart)var6).method3();      Sandwich 3
```

7. One possible solution appears below.

```
public void limitLeaves(int min) {
    overallRoot = limitLeaves(overallRoot, min);
}

private IntTreeNode limitLeaves(IntTreeNode root, int min) {
    if (root != null) {
        root.left = limitLeaves(root.left, min);
        root.right = limitLeaves(root.right, min);
        if (root.left == null && root.right == null && root.data <= min) {
            root = null;
        }
    }
    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;
}
```