Solution to CSE143X Section #12 Problems

Contest problems:

1.

Method Call	Value Returned
mystery1(6, 13)	6
mystery1(14, 10)	4
mystery1(37, 10)	7
mystery1(8, 2)	0
mystery1(50, 7)	1

2.

List before	List after
[1, 2, 3]	[0, 3, 2, 1]
[-1, -2, -3]	[0, 3, 2, 1]
[5, 0, 0, 4, -3, -2, 7, 0]	[0, 7, 2, 3, 4, 5]
[2, 4, -6, -8, 0, 10, 0, -12]	[0, 12, 10, 8, 6, 4, 2]
[0, 3, -2, 4, 0, 15]	[0, 15, 4, 2, 3]

3.

Method Call	Output Produced	Value Returned
showCollatz(1, 1);	1	4
showCollatz(2, 1);	2	1
showCollatz(4, 2);	4, 2	1
showCollatz(5, 2);	5, 16	8
showCollatz(7, 10);	7, 22, 11, 34, 17, 52, 26, 13, 40, 20	10

Section problems:

1.

Method Call	Output Produced
mystery2(1)	1
mystery2(4)	1, 2, 4
mystery2(16)	1, 2, 4, 8, 16
mystery2(30)	1, 3, 7, 15, 30
mystery2(100)	1, 3, 6, 12, 25, 50, 100

2.

Method Call	Value Returned
mystery3(6)	6
mystery3(17)	8
mystery3(259)	7
mystery3(977)	5
mystery3(-479)	-2

3.

Method Call	Value Returned
mystery4(8)	8
mystery4(74)	11
mystery4(-52)	7
mystery4(3052)	10
mystery4(82534)	22

4.

Method Call	Value Returned
mystery5(5, 7)	57
mystery5(12, 9)	1029
mystery5(-7, 4)	-74
mystery5(-23, -48)	2438
mystery5(128, 343)	132483

5.

Method Call	Output Produced
mystery6(4, 1)	4
mystery6(4, 2)	8, 4, 8
mystery6(8, 2)	16, 8, 16
mystery6(4, 3)	12, 8, 4, 8, 12
mystery6(3, 4)	12, 9, 6, 3, 6, 9, 12

6.

Method Call	Output Produced
mystery7(0)	*
mystery7(1)	[*]
mystery7(2)	([*])
mystery7(4)	([([*])])
mystery7(5)	[([([*])])]

7.

Method Call	Output Produced
mystery8(113)	113
mystery8(70)	140, 70
mystery8(42)	168, 84, 42
mystery8(30)	120, 60, 30
mystery8(10)	160, 80, 40, 20, 10

8.

Method Call	Output Produced
mystery9(7);	7
mystery9(38);	838
mystery9(194);	49194
mystery9(782);	28782
mystery9(3842);	2483842

9. One possible solution appears below.

```
public class Date implements Comparable<Date> {
    private int month;
    private int day;
    private int year;

    public Date(int month, int day, int year) {
        this.month = month;
        this.day = day;
        this.year = year;
    }

    public int getMonth() {
        return month;
    }

    public int getDay() {
        return day;
    }

    public int getYear() {
        return year;
    }
```

```
    public String toString() {
        String result;
        if (month < 10) {
            result = "0" + month + "/";
        } else {
            result = month + "/";
        }
        if (day < 10) {
            result += "0" + day;
        } else {
            result += day;
        }
        result += "/" + year;
        return result;
    }

    public int compareTo(Date other) {
        if (year != other.year) {
            return (year - other.year);
        } else if (month != other.month) {
            return (month - other.month);
        } else {
            return (day - other.day);
        }
    }
}
```

10. Two possible solutions appear below.

```
public class USCurrency implements Comparable<USCurrency> {
    private int dollars;
    private int cents;

    public USCurrency(int dollars, int cents) {
        int totalCents = 100 * dollars + cents;
        this.dollars = totalCents / 100;
        this.cents = totalCents % 100;
    }

    public int dollars() {
        return dollars;
    }

    public int cents() {
        return cents;
    }

    public String toString() {
        String s = Math.abs(dollars) + ".";
        if (Math.abs(cents) < 10) {
            s += "0" + Math.abs(cents);
        } else {
            s += Math.abs(cents);
        }
        if (dollars < 0 || cents < 0) {
            return "-$" + s;
        } else {
            return "$" + s;
        }
    }
```

```
    public USCurrency add(USCurrency other) {
        return new USCurrency(dollars + other.dollars,
                              cents + other.cents);
    }

    public USCurrency subtract(USCurrency other) {
        return new USCurrency(dollars - other.dollars,
                              cents - other.cents);
    }

    public int compareTo(USCurrency other) {
        return dollars * 100 + cents - other.dollars * 100 - other.cents;
    }
}

public class USCurrency implements Comparable<USCurrency> {
    private int totalCents;

    public USCurrency(int dollars, int cents) {
        totalCents = dollars * 100 + cents;
    }

    public int dollars() {
        return totalCents / 100;
    }

    public int cents() {
        return totalCents % 100;
    }

    public String toString() {
        int cents = Math.abs(totalCents);
        String s = cents / 100 + "." + cents % 100 / 10 + cents % 10;
        if (totalCents < 0) {
            return "-$" + s;
        } else {
            return "$" + s;
        }
    }

    public USCurrency add(USCurrency other) {
        return new USCurrency(dollars() + other.dollars(),
                              cents() + other.cents());
    }

    public USCurrency subtract(USCurrency other) {
        return new USCurrency(dollars() - other.dollars(),
                              cents() - other.cents());
    }

    public int compareTo(USCurrency other) {
        return totalCents - other.totalCents;
    }
}
```

11. One possible solution appears below.

```
public class ClockTime implements Comparable<ClockTime> {
    private int hours;
    private int minutes;
    private String amPm;

    public ClockTime(int hours, int minutes, String amPm) {
        this.hours = hours;
        this.minutes = minutes;
        this.amPm = amPm;
    }

    public int compareTo(ClockTime other) {
        if (!amPm.equals(other.amPm)) {
            return amPm.compareTo(other.amPm);
        } else if (hours != other.hours) {
            return hours % 12 - other.hours % 12;
        } else {
            return minutes - other.minutes;
        }
    }

    public int getHours() {
        return hours;
    }

    public int getMinutes() {
        return minutes;
    }

    public String getAmPm() {
        return amPm;
    }

    public String toString() {
        String result = hours + ":";
        if (minutes < 10) {
            result += "0" + minutes;
        } else {
            result += minutes;
        }
        result += " " + amPm;
        return result;
    }
}
```

12. 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 = getRating() - other.getRating();
        if (delta < 0) {
            return 1;
        } else if (delta > 0) {
            return -1;
        } else // delta == 0 {
            return other.reviews - reviews;
        }
    }
}
```