CSE 142 Sample Final Exam #2

1. Expressions

For each expression in the left-hand column, indicate its value in the right-hand column.
List a constant of appropriate type (e.g., `7.0` for a `double`, `true` or `false` for a `boolean`, `Strings` in `""` quotes).

Expression	Value
`12 / 5 + 8 / 4`	____________
`2.5 * 2 + 17 / 4`	____________
`4.0 > 2.5 || (!(5 < 2) && 1.1 > 0)`	____________
`21 / 2 + "7 % 3" + 17 % 4`	____________
`46 / 3 / 2.0 / 3 * 4 / 5`	____________

2. Array Mystery

Consider the following method:

```
public static int mystery(int[] array) {
    int x = 0;
    for (int i = 0; i < array.length - 1; i++) {
        if (array[i] > array[i + 1]) {
            x++;
        }
    }
    return x;
}
```

In the left-hand column below are specific arrays of integers. Indicate in the right-hand column what value would be returned by method `mystery` if the integer array in the left-hand column is passed as its parameter.

Array	Value Returned

`int[] a1 = {8};`
`mystery(a1);`
	________

`int[] a2 = {14, 7};`
`mystery(a2);`
	________

`int[] a3 = {7, 1, 3, 2, 0, 4};`
`mystery(a3);`
	________

`int[] a4 = {10, 8, 9, 5, 6};`
`mystery(a4);`
	________

`int[] a5 = {8, 10, 8, 6, 4, 2};`
`mystery(a5);`
	________

3. Inheritance Mystery

Assume that the following classes have been defined:

```
public class Pen extends Sock {
    public void method1() {
        System.out.print("pen 1  ");
    }
}

public class Lamp {
    public void method1() {
        System.out.print("lamp 1  ");
    }

    public void method2() {
        System.out.print("lamp 2  ");
    }

    public String toString() {
        return "lamp";
    }
}

public class Book extends Sock {
    public void method2() {
        System.out.print("book 2  ");
        super.method2();
    }
}

public class Sock extends Lamp {
    public void method1() {
        System.out.print("sock 1  ");
        method2();
    }

    public String toString() {
        return "sock";
    }
}
```

Given the classes above, what output is produced by the following code?

```
Lamp[] elements = {new Book(), new Pen(), new Lamp(), new Sock()};
for (int i = 0; i < elements.length; i++) {
    System.out.println(elements[i]);
    elements[i].method1();
    System.out.println();
    elements[i].method2();
    System.out.println();
    System.out.println();
}
```

4. File Processing

Write a static method named `wordStats` that accepts as its parameter a `Scanner` holding a sequence of words and that reports the total number of words and the average word length. For example, suppose the `Scanner` is scanning an input source that contains the following words:

```
To be or not to be, that is the question.
```

For the purposes of this problem, we will use whitespace to separate words. That means that some words include punctuation, as in `"be,"`. (This is the same definition that the `Scanner` uses for tokens.) For the input above, your method should produce exactly the following output:

```
Total words    = 10
Average length = 3.2
```

5. File Processing

Write a static method named `flipLines` that accepts as its parameter a `Scanner` for an input file and that writes to the console the same file's contents with successive pairs of lines reversed in order. For example, if the input file contains the following text:

```
Twas brillig and the slithy toves
did gyre and gimble in the wabe.
All mimsey were the borogroves,
and the mome raths outgrabe.

"Beware the Jabberwock, my son,
the jaws that bite, the claws that catch,
Beware the JubJub bird and shun
the frumious bandersnatch."
```

The program should print the first pair of lines in reverse order, then the second pair in reverse order, then the third pair in reverse order, and so on. Therefore your method should produce the following output to the console:

```
did gyre and gimble in the wabe.
Twas brillig and the slithy toves
and the mome raths outgrabe.
All mimsey were the borogroves,
"Beware the Jabberwock, my son,

Beware the JubJub bird and shun
the jaws that bite, the claws that catch,
the frumious bandersnatch."
```

Notice that a line can be blank, as in the third pair. Also notice that an input file can have an odd number of lines, as in the one above, in which case the last line is printed in its original position. You may not make any assumptions about how many lines are in the Scanner.

6. **Array Programming**
Write a static method named `minGap` that accepts an integer array as a parameter and returns the minimum 'gap' between adjacent values in the array. The gap between two adjacent values in a array is defined as the second value minus the first value. For example, suppose a variable called `array` is an array of integers that stores the following sequence of values.

```
int[] array = {1, 3, 6, 7, 12};
```

The first gap is 2 (3 - 1), the second gap is 3 (6 - 3), the third gap is 1 (7 - 6) and the fourth gap is 5 (12 - 7). Thus, the call of `minGap(array)` should return `1` because that is the smallest gap in the array. Notice that the minimum gap could be a negative number. For example, if array stores the following sequence of values:

```
(3, 5, 11, 4, 8)
```

The gaps would be computed as 2 (5 - 3), 6 (11 - 5), -7 (4 - 11), and 4 (8 - 4). Of these values, -7 is the smallest, so it would be returned.

This gap information can be helpful for determining other properties of the array. For example, if the minimum gap is greater than or equal to 0, then you know the array is in sorted (nondecreasing) order. If the gap is greater than 0, then you know the array is both sorted and unique (strictly increasing).

If you are passed an array with fewer than 2 elements, you should return 0.

7. **Array Programming**
Write a static method named `longestSortedSequence` that accepts an array of integers as a parameter and that returns the length of the longest sorted (nondecreasing) sequence of integers in the array. For example, if a variable named `array` stores the following values:

```
int[] array = {3, 8, 10, 1, 9, 14, -3, 0, 14, 207, 56, 98, 12};
```

then the call of `longestSortedSequence(array)` should return `4` because the longest sorted sequence in the array has four values in it (the sequence -3, 0, 14, 207). Notice that sorted means nondecreasing, which means that the sequence could contain duplicates. For example, if the array stores the following values:

```
int[] array2 = {17, 42, 3, 5, 5, 5, 8, 2, 4, 6, 1, 19}
```

Then the method would return `5` for the length of the longest sequence (the sequence 3, 5, 5, 5, 8). Your method should return 0 if passed an empty array.

8. Critters

Write a class `Yak` that extends the `Critter` class from the Critters assignment.
`Yak` objects should move as follows: First pick a random direction and move one step in that direction, then pick a random direction and move two steps in that direction, then pick a random direction and move three steps in that direction, then pick a random direction and move four steps in that direction, and so on, each time picking a random direction and going in that direction one further than it went in the previous direction. Each direction should be equally likely. All other `Yak` behavior uses the `Critter` defaults.

You may add anything needed (fields, other methods) to implement this behavior appropriately.

9. Classes and Objects

Write a method named addDays that will be placed inside the Date class from the Birthday/Date assignment, as shown at right.

The addDays method accepts an integer as a parameter and shifts the date represented by the Date object forward by that many days. You may assume the value passed is non-negative.

For example, if the following Date is declared in client code:

```
Date d = new Date(9, 19);
```

The following calls to the addDays method would modify the Date object's state as indicated in the comments. Remember that Date objects do not store the year; the date after December 31st is January 1st.

```
d.addDays(1);      // d now represents Sep. 20
d.addDays(1);      // d now represents Sep. 21
d.addDays(5);      // d now represents Sep. 26
d.addDays(10);     // d now represents Oct. 6
d.addDays(80);     // d now represents Dec. 25
d.addDays(10);     // d now represents Jan. 4
                   // (of the next year)
d.addDays(1000);   // d now represents Oct. 1
                   // (over two years later)
```

```
public class Date {
    private int month;
    private int day;

    public Date(int m, int d)

    public int getDay()

    public int getMonth()

    public int daysInMonth()

    public void nextDay()

    // your method would go here

}
```

Solutions

1.

```
Expression                              Value
-----------------------------------------------
12/ 5 + 8 / 4                           4
2.5 * 2 + 17 / 4                        9.0
4.0 > 2.5 || (!(5 < 2) && 1.1 > 0)      true
21 / 2 + "7 % 3" + 17 % 4               "107 % 31"
46 / 3 / 2.0 / 3 * 4/5                  2.0
```

2.

```
Array                               Value returned
-----------------------------------------------
{8}                                     0
{14, 7}                                 1
{7, 1, 3, 2, 0, 4}                      3
{10, 8, 9, 5, 6}                        2
{8, 10, 8, 6, 4, 2}                     4
```

3.

```
sock
sock 1  book 2  lamp 2
book 2  lamp 2

sock
pen 1
lamp 2

lamp
lamp 1
lamp 2

sock
sock 1  lamp 2
lamp 2
```

4.

```
public static void wordStats(Scanner input) {
    int count = 0;
    int sumLength = 0;

    while (input.hasNext()) {
        String next = input.next();
        count++;
        sumLength += next.length();
    }

    double average = (double) sumLength / count;
    System.out.println("Total words    = " + count);
    System.out.println("Average length = " + average);
}
```

5.

```
public static void flipLines(Scanner input) {
    while (input.hasNextLine()) {
        String first = input.nextLine();
        if (input.hasNextLine()) {
            String second = input.nextLine();
            System.out.println(second);
        }
        System.out.println(first);
    }
}
```

6.

```
public static int minGap(int[] list) {
    if (list.length < 2) {
        return 0;
    } else {
        int min = list[1] - list[0];
        for (int i = 2; i < list.length; i++) {
            int gap = list[i] - list[i - 1];
            if (gap < min) {
                min = gap;
            }
        }
        return min;
    }
}
```

7.

```
public static int longestSortedSequence(int[] list) {
    if (list.length == 0) {
        return 0;
    }

    int max = 1;
    int count = 1;
    for (int i = 1; i < list.length; i++) {
        if (list[i] >= list[i - 1]) {
            count++;
        } else {
            count = 1;
        }

        if (count > max) {
            max = count;
        }
    }
    return max;
}
```

8.

```
public class Yak extends Critter {
    private Random rand;
    private int steps;
    private int max;
    private int direction;

    public Yak() {
        rand = new Random();
        steps = 0;
        max = 0;
        direction = 0;
    }

    public Direction getMove() {
        // Pick a new direction and re-set the steps counter
        if (steps == max) {
            steps = 0;
            max++;
            direction = rand.nextInt(4);
        }

        steps++;
        if (direction == 0) {
            return Direction.NORTH;
        } else if (direction == 1) {
            return Direction.SOUTH;
        } else if (direction == 2) {
            return Direction.EAST;
        } else {  // direction == 3
            return Direction.WEST;
        }
    }

}
```

9. Four solutions are shown.

```
public void addDays(int days) {
    for (int i = 0; i < days; i++) {
        day++;
        if (day > daysInMonth()) {
            day = 1;
            month++;
        }
        if (month > 12) {
            month = 1;
            day = 1;
        }
    }
}

public void addDays(int days) {
    while (day + days > daysInMonth()) {
        days -= daysInMonth();
        month++;
        if (month > 12) {
            month = 1;
        }
    }
    day += days;
}

public void addDays(int days) {
    day += days;
    while (day > 365) {
        day -= 365;
    }
    while (day > daysInMonth()) {
        day -= daysInMonth();
        month++;
        if (month > 12) {
            month = 1;
        }
    }
}

public void addDays(int days) {
    for (int i = 0; i < days; i++) {
        nextDay();
    }
}
```