CSE 142, Summer 2010
Midterm Exam, Friday, July 30, 2010

Name:	______________________________		
Section:	____________	**TA:**	____________
Student ID #:	____________		

- You have 60 minutes to complete this exam.
  You may receive a deduction if you keep working after the instructor calls for papers.
- This exam is open-book/notes. You may not use any calculators or other computing devices.
- Code will be graded on proper behavior/output and not on style, unless otherwise indicated.
- Do not abbreviate code, such as "ditto" marks or dot-dot-dot ... marks.
  The *only* abbreviations that *are* allowed for this exam are:
  - `S.o.p` for `System.out.print`,
  - `S.o.pln` for `System.out.println`, and
  - `S.o.pf` for `System.out.printf`.
- You do not need to write `import` statements in your code.
- If you enter the room, you must turn in an exam before leaving the room.
- You must show your Student ID to a TA or instructor for your exam to be accepted.

Good luck!

Score summary: (for grader only)

Problem	**Description**	**Earned**	**Max**
1	Expressions		10
2	Parameter Mystery		15
3	If/Else Simulation		10
4	While Loop Simulation		10
5	Assertions		15
6	Programming		15
7	Programming		15
8	Programming		10
TOTAL	**Total Points**		**100**

1. Expressions

For each expression at left, indicate its value in the right column. List a value of appropriate type and capitalization. e.g., `7` for an `int`, `7.0` for a `double`, `"hello"` for a `String`, `true` or `false` for a `boolean`.

Expression	Value
`11 - 7 * 2 + 3 * 4`	______________________________
`10 / (5 / 2) + 1.5 * 6 / 2`	______________________________
`115 / 10 + 115 % 10 + 11 / 7 * 3.5`	______________________________
`"0" + 10 * 5 + "0" + 5 + 6`	______________________________
`(4 <= 9 / 2) && !(2 != 10)`	______________________________

2. Parameter Mystery

At the bottom of the page, write the output produced by the following program, as it would appear on the console.

```
public class ParameterMystery {
    public static void main(String[] args) {
        String scarlett = "mustard";
        String suspect = "peacock";
        String lounge = "ballroom";
        String pipe = "study";
        String dagger = "pipe";
        String weapon = "dagger";

        clue(weapon, suspect, pipe);
        clue(scarlett, pipe, suspect);
        clue(dagger, "lounge", scarlett);
        clue(lounge, dagger, "dagger");
        clue(pipe, "miss " + scarlett, lounge);
    }

    public static void clue(String suspect, String room, String weapon) {
        System.out.println(room + " in the " + weapon + " with the " + suspect);
    }
}
```

3. If/Else Simulation

For each call below to the following method, write the output that is produced, as it would appear on the console:

```
public static void ifElseMystery(int a, int b) {
    if (a % 10 == 0) {
        b *= 10;
        a += 10;
    }
    if (a > b) {
        b = a;
    } else if (a == b) {
        b++;
    }
    System.out.println(a + " " + b);
}
```

Method Call	Output
`ifElseMystery(20, 8);`	____
`ifElseMystery(30, 30);`	____
`ifElseMystery(4, 3);`	____
`ifElseMystery(30, 4);`	____
`ifElseMystery(7, 7);`	____

4. While Loop Simulation

For each call below to the following method, write the output that is produced, as it would appear on the console:

```java
public static void whileMystery(int i, int j) {
    int k = 0;
    while (i < j && k < j) {
        i = i + k;
        j--;
        k++;
        System.out.print(i + ", ");
    }

    System.out.println(k);
}
```

Method Call	Output
`whileMystery(3, 5);`	______________________________________
`whileMystery(5, 3);`	______________________________________
`whileMystery(-3, 6);`	______________________________________
`whileMystery(2, 12);`	______________________________________
`whileMystery(-9, 10);`	______________________________________

5. Assertions

For each of the five points labeled by comments, identify each of the assertions in the table below as either being *always* true, *never* true, or *sometimes* true / sometimes false. (You may abbreviate them as A, N, or S.)

```
// Reads many integers from Scanner; returns the second smallest
// integer read.
public static int secondSmallest(Scanner console) {
    int num = console.nextInt();
    int first = num;
    int second = num;

    // Point A
    while (num >= 0) {
        // Point B
        if (num < first) {
            second = first;
            first = num;
            // Point C
        } else if (num < second) {
            second = num;
        }

        // Point D
        num = console.nextInt();
    }

    // Point E
    return second;
}
```

	num < 0	first < second	num >= second
Point A			
Point B			
Point C			
Point D			
Point E			

6. Programming

Write a static method `speedingTicket` that decides whether a given driver should be given a speeding ticket from a police officer. The method accepts three parameters: the driver's car *speed* in miles per hour, as an integer; the current *speed limit*, as an integer; and whether or not the police officer has eaten a *donut* today, as a `true`/`false` value. (A police officer that has eaten a donut is happy, so he/she is less likely to give the driver a ticket.)

Your method should return `true` if the driver should receive a speeding ticket, and `false` if not.
A driver should be given a speeding ticket if any of the following conditions are met:

- The officer has eaten a donut (`true`) and the driver's speed is at least 10 mph over the speeding limit.
- The officer has not eaten a donut (`false`) and the driver's speed is at least 5 mph over **or under** the limit.
- The driver is going 100 mph or faster, regardless of the speed limit or donut status.

You may assume that the integers passed as parameters will be non-negative.
Here are some example calls to the method and their resulting return values:

Call	Value Returned
`speedingTicket(58, 55, false)`	`false`
`speedingTicket(51, 55, false)`	`false`
`speedingTicket(42, 35, false)`	`true`
`speedingTicket(30, 35, false)`	`true`
`speedingTicket(45, 30, true)`	`true`
`speedingTicket(49, 40, true)`	`false`
`speedingTicket(22, 30, true)`	`false`
`speedingTicket(102, 15, false)`	`true`
`speedingTicket(101, 98, true)`	`true`

7. Programming

It is said that all dogs hear when people talk to them are their names (see cartoon by Gary Larson to the right). Write a static method named `dogHears` that prints lines of what a dog hears. The method accepts two parameters: a *dog's name*, as a string, and the *number of lines*, as an integer. Each line contains a randomly chosen number of words, between 2 - 10 words inclusive.

What we say to dogs
Okay, Ginger! I've had it! You stay out of the garbage! Understand, Ginger? Stay out of the garbage, or else!
What they hear
blah blah GINGER blah blah blah blah blah blah blah blah GINGER blah blah blah blah blah blah...

Each word on a given line is chosen randomly to be one of two choices: the dog's name, with 25% probability (1/4 likelihood); or the word "blah", with 75% probability (3/4 likelihood). This does not mean that the dogs name will print exactly every fourth time; just that a given word has a 1/4 random chance of being the dog's name. Multiple calls to `dogHears` with the same parameters will likely result in different output for each call.

If 0 or less is passed for the number of lines, no output should be produced.

The following table lists some calls to `dogHears`:

Call	`dogHears("Ginger", 7);`	`dogHears("Kona", 4);`

Example Output	`Ginger blah blah blah blah blah blah`
	`blah blah blah blah blah blah`
	`blah blah Ginger blah blah`
	`Ginger blah blah Ginger blah blah blah blah blah`
	`blah Ginger blah blah blah blah Ginger blah`
	`blah Ginger`
	`blah blah blah blah blah blah Ginger blah blah Ginger`
		`blah blah blah`
		`Kona Kona blah blah blah Kona`
		`blah blah Kona blah`
		`blah Kona blah`

8. Programming

Write a method `highLow` that takes a *number*, an integer, as a parameter and returns whether or not the number has alternating "high" and "low" digits. 0 through 4 are "low" digits and 5 through 9 are "high" digits. Your method should return `true` if the number passed alternates between "high" and "low" digits, and `false` if not. You may assume the number passed is positive. If the number passed consists of a single digit, `highLow` should return `true`.

Note: `highLow` returns `true` if the number alternates starting with a "high" digit or starting with a "low" digit. What is important is that the digits alternate. For example, both 9292 and 2929 passed to `highLow` should return `true`.

Here are some example calls to the method and their resulting return values:

Call	**Value Returned**
`highLow(1918193)`	`true`
`highLow(7283)`	`true`
`highLow(3827)`	`true`
`highLow(9388)`	`false`
`highLow(895151)`	`false`
`highLow(707)`	`true`
`highLow(44)`	`false`
`highLow(45)`	`true`
`highLow(5)`	`true`