CSE 142, Autumn 2009
Midterm Exam, Friday, November 6, 2009

Name: _____

Section: _____ **TA:** _____

Student ID #: _____ **Seat:** _____

- You have 50 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		12
3	If/Else Simulation		13
4	While Loop Simulation		15
5	Assertions		15
6	Programming		20
7	Programming		15
TOTAL	**Total Points**		**100**

(This page intentionally left blank.)

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
`1 + 2 * 3 * 4 - 5 * 2`	_____
`2 + "(int)2.0" + 2 * 2 + 2`	_____
`15 % 3 == 0 && !(3 > 2 && 1 > 3)`	_____
`1 / 2 + -(157 / 10 / 10.0) + 9.0 * 1 / 2`	_____
`24 % 5 + 9 % (6 % 4)`	_____

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 literal = "8";
        String brace = "semi";
        String paren = brace;
        String semi = "brace";
        String java = "42";

        param(java, brace, semi);
        param(literal, paren, java);
        param(brace, semi, "literal");
        param("cse", literal + 4, "1");
    }

    public static void param(String semi, String java, String brace) {
        System.out.println(java + " missing a " + brace + " and " + semi);
    }
}
```

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, int c) {
    if (a < b && a < c) {
        a = a + c;
        c++;
    } else if (a >= b) {
        a = a - b;
        b--;
    }
    if (a >= b && a >= c) {
        a++;
        c++;
    }

    System.out.println(a + " " + b + " " + c);
}
```

Method Call	Output
`ifElseMystery(2, 10, 3);`	_____
`ifElseMystery(8, 6, 1);`	_____
`ifElseMystery(4, 6, 7);`	_____
`ifElseMystery(20, 5, 5);`	_____

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:

```
public static void whileMystery(int x, int y) {
    int z = 0;
    while (x % y != 0) {
        x = x / y;
        z++;
        System.out.print(x + ", ");
    }

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

Method Call	Output
`whileMystery(25, 2);`	_____
`whileMystery(10345, 10);`	_____
`whileMystery(63, 2);`	_____

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.)

```
public static int count(int n) {
    int even = 0;
    int odd = 0;

    // Point A
    while (n != 0 && even <= odd) {
        if (n % 2 == 0) {
            even++;
            // Point B
        } else {
            // Point C
            odd++;
        }

        n = n / 2;
        // Point D
    }

    // Point E
    return even - odd;
}
```

	n == 0	even <= odd	n % 2 == 0
Point A			
Point B			
Point C			
Point D			
Point E			

6. Programming

Write a static method named `longestName` that reads names typed by the user and prints the longest name (the name that contains the most characters) in the format shown below. Your method should accept a console `Scanner` and an integer *n* as parameters and should then prompt for *n* names.

The longest name should be printed with its first letter capitalized and all subsequent letters in lowercase, regardless of the capitalization the user used when typing in the name.

If there is a tie for longest between two or more names, use the tied name that was typed earliest. Also print a message saying that there was a tie, as in the right log below. It's possible that some shorter names will tie in length, such as DANE and Erik in the left log below; but don't print a message unless the tie is between the longest names.

You may assume that *n* is at least 1, that each name is at least 1 character long, and that the user will type single-word names consisting of only letters. The following table shows two sample calls and their output.

Call	`Scanner console = new Scanner(System.in);` `longestName(console, 5);`	`Scanner console = new Scanner(System.in);` `longestName(console, 7);`

Output	`name #1? roy`
	`name #2? DANE`
	`name #3? Erik`
	`name #4? sTeFaNiE`
	`name #5? LaurA`
	`Stefanie's name is longest`
		`name #1? TrEnt`
		`name #2? rita`
		`name #3? JORDAN`
		`name #4? craig`
		`name #5? leslie`
		`name #6? YUKI`
		`name #7? TaNnEr`
		`Jordan's name is longest`
		`(There was a tie!)`

(extra writing space for Problem #6)

7. Programming

Write a static method named `largerDigits` that accepts two integer parameters *a* and *b* and returns a new integer *c* where each digit of *c* gets its value from the larger of *a*'s and *b*'s digit in the same place. That is, the ones digit of *c* is the larger of the ones digit of *a* and the ones digit of *b*, and the tens digit of *c* is the larger of the tens digit of *a* and the tens digit of *b*, and so on. You may assume that *a* and *b* are positive integers (greater than 0).

For example, suppose *a* is 603452384 and *b* is 921782. Their digits would be combined as follows to produce *c*:

```
a        603452380
b           920784
------------------
c           952784      (return value)
```

Notice that if a particular digit place is absent from one number or the other, such as the 603 at the start of *a* above, no digit is carried over to *c*. The following table lists some more calls to your method and their expected return values:

Call	Value Returned
`largerDigits(172, 312)`	`372`
`largerDigits(21, 3)`	`3`
`largerDigits(90, 38906735)`	`95`
`largerDigits(56002, 123321)`	`56322`
`largerDigits(11223, 4466)`	`4466`
`largerDigits(12345, 12345)`	`12345`
`largerDigits(1, 34892)`	`2`

Hint: If you are building a result number, you may need to use `Math.pow` or accumulate a multiplier with each digit.

You may not use a `String` to solve this problem.