CSE 142, Winter 2008
Midterm Exam, Friday, February 15, 2008

Name: _____

Section: _____ **TA:** _____

Student ID #: _____

Rules:

- You have 50 minutes to complete this exam.
  You may receive a deduction if you keep working after the instructor calls for papers.
- This test is open-book/notes.
- You may not use any computing devices of any kind including calculators.
- Unless otherwise indicated, your code will be graded on proper behavior/output, not on style.
- You do not need to write any `import` statements in your exam code.
- Please *do not* abbreviate any code, such as writing "ditto" marks or dot-dot-dot marks . . .

  The <u>one abbreviation that *is* allowed</u> for this exam is to write `S.o.p` for `System.out.println`.

- If you enter the room, you must turn in an exam and will not be permitted to leave without doing so.
- You must show your Student ID to a TA or instructor for your submitted exam to be accepted.

Good luck!

Score summary: (for grader use only)

Problem	Description	Earned	Max
1	Expressions		15
2	Parameter Mystery		20
3	While Loop Simulation		15
4	Assertions		15
5	Programming		20
6	Programming		15
X	Extra Credit		+1
TOTAL	**Total Points**		**100**

1. Expressions (15 points)

For each expression in the left-hand column, indicate its value in the right-hand column.
Be sure to list a constant 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
`4 + 3 * 2 - 1`	_____
`1 + ( 2 + " ( 3 + " + 4 + " 5 ) " ) + 6`	_____
`100 % 30 % 4 + 5 % 2`	_____
`!(1 + 1 >= 2) || 10 / 3 == 3`	_____
`2.0 * 2.2 + -23 / 2 / 2.0`	_____
`10 > 5 && (2 > 3 || 5 <= 4 + 2)`	_____

2. Parameter Mystery (20 points)

At the bottom of the page, write the output produced by the following program, as it would appear on the console.
(Though the program uses words related to chess, the output does not necessarily relate to real chess game rules.)

```
public class ParameterMystery {
    public static void main(String[] args) {
        String knight = "rook";
        String king = "pawn";
        String bishop = "knight";
        String pawn = "queen";
        String rook = bishop;

        chess(knight, king, bishop);
        chess(bishop, knight, king);
        chess(king, pawn, "queen");
        bishop = "king";
        chess("bishop", rook, knight);
        chess(pawn, bishop, bishop);
    }

    public static void chess(String b, String c, String a) {
        System.out.println("A " + a + " and a " + b + " beats a " + c);
    }
}
```

3. While Loop Simulation (15 points)

For each call below to the following method, write the value that is returned:

```
public static int whileMystery(int a) {
    int b = 1;
    int c = 0;

    while (b <= a) {
        if (b % 2 == 0) {
            c++;
        }

        b = b * 10;
        c++;
    }

    return c + 1;
}
```

Method Call	Value Returned
`whileMystery(2)`	_____
`whileMystery(84)`	_____
`whileMystery(4097)`	_____
`whileMystery(388)`	_____
`whileMystery(23980)`	_____

4. Assertions (15 points)

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.

```
public static int dice(Random rand) {
    int count = 0;
    int even = 0;

    // Point A
    while (count < 10 && even < 3) {
        // Point B
        count++;
        int roll = rand.nextInt(6) + 1;
        if (roll % 2 == 0) {
            // Point C
            even++;

            // Point D
        }
    }

    // Point E
    return count;
}
```

Fill in each box of the the table below with one of the following words: ALWAYS, NEVER or SOMETIMES. (You may abbreviate these choices as A, N, and S.)

	`count < 10`	`even > 0`	`even < 3`
Point A			
Point B			
Point C			
Point D			
Point E			

5. Programming (20 points)

In this question, we'll address the following problem: Can a cash register containing a given amount of pennies (1-cent coins) and a given amount of nickels (5-cent coins) give a customer a given exact amount of cents of change? For example, if there are 3 pennies and 5 nickels in the cash register, is it possible to give exactly 19 cents of change? (No.) If there are 2 pennies and 7 nickels in the register, is it possible to give exactly 26 cents of change? (Yes.)

Write a static method named `canMakeChange` that accepts three integer parameters representing the number of pennies in the cash register, the number of nickels in the cash register, and the desired amount of change to make. The method should return `true` if the coins in the register can produce this exact amount of change, and `false` if not. The coins in the register must be able to *exactly* produce the desired amount of change in order to return `true`; for example, if the register contains 0 pennies and 100 nickels, it is *not* able to exactly produce 8 cents of change.

The following are several sample calls to your method and the values they should return. You may assume that no negative parameter values are passed, but otherwise your method should work with any values passed.

Call		Value Returned
`canMakeChange(3, 4, 12)`	`// 3 pennies, 4 nickels, 12c change?`	`true`
`canMakeChange(1, 5, 26)`	`// 1 penny, 5 nickels, 26c change?`	`true`
`canMakeChange(24, 2, 31)`	`// 24 pennies, 2 nickels, 31c change?`	`true`
`canMakeChange(87, 19, 134)`	`// 87 pennies, 19 nickels, 134c change?`	`true`
`canMakeChange(0, 0, 0)`	`// 0 pennies, 0 nickels, 0c change?`	`true`
`canMakeChange(1, 1, 9)`	`// 1 penny, 1 nickel, 9c change?`	`false`
`canMakeChange(2, 7, 8)`	`// 2 pennies, 7 nickels, 8c change?`	`false`
`canMakeChange(4, 3, 39)`	`// 4 pennies, 3 nickels, 39c change?`	`false`
`canMakeChange(3, 80, 14)`	`// 3 pennies, 80 nickels, 14c change?`	`false`

6. Programming (15 points)

Write a static method named `meetings` that schedules meetings of varying random lengths, starting from a given time, until a point in time where a meeting ends **within 5 minutes of an hour mark**. Your method should accept two parameters representing the starting hour and starting minute. The method should repeatedly produce random meeting lengths **between 15 and 30 minutes inclusive**, printing what the time would be after that meeting is over. You should stop when a meeting ends during the first or last 5 minutes of a given hour, that is, if the minutes hand of the clock is on 5 or less or 55 or higher. At least one meeting should always be scheduled regardless of the start time.

The following table shows three example calls to your method and their resulting output. To simplify the output in cases with minutes < 10, we'll print times in a "2h 17m" format instead of the more familiar "2:17".

Call	`meetings(2, 17);`	`meetings(10, 14);`	`meetings(7, 3);`

Output	`2h 17m start time`
	`2h 40m after 23-min meeting`
	`3h 7m after 27-min meeting`
	`3h 36m after 29-min meeting`
	`3h 51m after 15-min meeting`
	`4h 9m after 18-min meeting`
	`4h 33m after 24-min meeting`
	`5h 0m after 27-min meeting`
		`10h 14m start time`
		`10h 39m after 25-min meeting`
		`10h 54m after 15-min meeting`
		`11h 12m after 18-min meeting`
		`11h 30m after 18-min meeting`
		`11h 47m after 17-min meeting`
		`12h 6m after 19-min meeting`
		`12h 22m after 16-min meeting`
		`12h 41m after 19-min meeting`
		`1h 9m after 28-min meeting`
		`1h 35m after 26-min meeting`
		`2h 5m after 30-min meeting`
			`7h 3m start time`
			`7h 18m after 15-min meeting`
			`7h 38m after 20-min meeting`
			`7h 57m after 19-min meeting`

Note that a meeting can cause the current time to wrap into the next hour, and that the next hour after 12 is 1. Do not worry about AM vs. PM time for this problem. You may assume the parameter values passed will be legal values (starting hour between 1 and 12, starting minute between 0 and 59).

X. Extra Credit (+1 point)
Answer either one of the following two questions to get +1 point. (You don't get +2 for doing both.)

a) The acronym for our lab is "IPL". Based on your time there, what do you think "IPL" is short for?

(Any phrase you write that begins with I, P, and L will get the +1 point.
The point isn't whether you know the actual answer; it's to come up with your own silly set of words.)

or

b) How *would* you have spent your Valentine's Day if you hadn't been studying for this exam?
Briefly write or draw what you would rather have been doing!

(As long as we can tell what it is, any phrase or picture will receive the bonus point. Nothing X-rated, please!)