CSE 142, Autumn 2010
Midterm Exam, Friday, November 5, 2010

Name: _____

Section: _____ **TA:** _____

Student ID #: _____

- 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		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
`12 / 3 + 5 + 3 * -2`	_____
`1 + 1 + "(1 + 1)" + 1 + 1`	_____
`13 / 2 - 38 / 5 / 2.0 + (15 / 10.0)`	_____
`11 < 3 + 4 || !(5 / 2 == 2)`	_____
`20 % 6 + 6 % 20 + 6 % 6`	_____

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) {
        int a = 5;
        int b = 1;
        int c = 3;
        int three = a;
        int one = b + 1;

        axiom(a, b, c);
        axiom(c, three, 10);
        axiom(b + c, one + three, a + one);
        a++;
        b = 0;
        axiom(three, 2, one);
    }

    public static void axiom(int c, int b, int a) {
        System.out.println(a + " + " + c + " = " + b);
    }
}
```

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 x, int y) {
    if (x == y) {
        x = x + 11;
    } else if (x > 2 * y) {
        x = 0;
    }
    if (x == 0 || y > x) {
        x = x + 2;
        y = y + 2;
    }

    System.out.println(x + " " + y);
}
```

Method Call	Output
`ifElseMystery(5, 5);`	_____
`ifElseMystery(18, 4);`	_____
`ifElseMystery(3, 6);`	_____

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) {
    while (x > 0 && y > 0) {
        x = x - y;
        y--;
        System.out.print(x + " ");
    }

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

Method Call	Output
`whileMystery(7, 5);`	_____
`whileMystery(20, 4);`	_____
`whileMystery(40, 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.)

```
public static int stuff(Random r, int m) {
    int c = 0;
    int t = 0;
    int d = r.nextInt(m);

    // Point A

    while (c <= 3) {
        // Point B

        d = r.nextInt(6) + 1;
        if (d <= m) {
            c++;
            // Point C

        } else {
            c = 0;
            // Point D

        }
        t++;
    }

    // Point E

    return t;
}
```

	c > 3	d <= m	c == 0
Point A			
Point B			
Point C			
Point D			
Point E			

6. Programming

Write a static method named `xo` that accepts an integer *size* as a parameter and prints a square of *size* by *size* characters, where all characters are "o" except that an "x" pattern of "x" characters has been drawn from the corners of the square. In other words, on the first line, the first and last characters are "x"; on the second line, the second and second-from-last characters are "x"; and so on. If 0 or less is passed for the size, no output should be produced.

The following table lists some calls to your method and their expected output:

Call	`xo(5);`	`xo(8);`	`xo(3);`	`xo(1);`	`xo(0);`	`xo(12);`	`xo(11);`
Example Output	x○○○x ○x○x○ ○○x○○ ○x○x○ x○○○x	x○○○○○○x ○x○○○○x○ ○○x○○x○○ ○○○xx○○○ ○○○xx○○○ ○○x○○x○○ ○x○○○○x○ x○○○○○○x	x○x ○x○ x○x	x		x○○○○○○○○○○x ○x○○○○○○○○x○ ○○x○○○○○○x○○ ○○○x○○○○x○○○ ○○○○x○○x○○○○ ○○○○○xx○○○○○ ○○○○○xx○○○○○ ○○○○x○○x○○○○ ○○○x○○○○x○○○ ○○x○○○○○○x○○ ○x○○○○○○○○x○ x○○○○○○○○○○x	x○○○○○○○○○○x ○x○○○○○○○○x○ ○○x○○○○○○x○○ ○○○x○○○○x○○○ ○○○○x○○x○○○○ ○○○○○x○○○○○ ○○○○x○○x○○○○ ○○○x○○○○x○○○ ○○x○○○○○○x○○ ○x○○○○○○○○x○ x○○○○○○○○○○x

7. Programming

Write a static method named `anglePairs` that accepts three angles (integers), measured in degrees, as parameters and returns whether or not there exists both complementary and supplementary angles amongst the three angles passed. Two angles are *complementary* if their sum is exactly 90 degrees; two angles are *supplementary* if their sum is exactly 180 degrees. Therefore, the method should return `true` if any two of the three angles add up to 90 degrees and also any two of the three angles add up to 180 degrees; otherwise the method should return `false`. You may assume that each angle passed is non-negative.

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

Call	Value Returned
`anglePairs(0, 90, 180)`	`true`
`anglePairs(45, 135, 45)`	`true`
`anglePairs(177, 87, 3)`	`true`
`anglePairs(120, 60, 30)`	`true`
`anglePairs(35, 60, 30)`	`false`
`anglePairs(120, 60, 45)`	`false`
`anglePairs(45, 90, 45)`	`false`
`anglePairs(180, 45, 45)`	`false`

8. Programming

Write a static method named `baseball` that takes a `Scanner` representing the console as a parameter and plays an interactive baseball scoring game with the user, returning an integer representing which team won the game.

Baseball is a sport where teams play a series of *innings* against each other. Each team scores a certain number of runs (points) in each inning. A baseball game ends when one of the following conditions is met:

- After 9 innings are finished, if one team has more total runs than the other, the team with more runs wins.
- After 9 innings are finished, if the teams are tied (if they have the same number of total runs), we keep playing one more full inning at a time until the teams are not tied any more.
- After any inning, if one team is ahead by 10 or more runs, the game is called and the team with more runs wins.

Your method should repeatedly prompt the user, once per inning, to enter the number of runs that each team scored during each inning. The user will type in the first team's runs for that inning, followed by the second team's runs for that inning, separated by whitespace. Your method should stop prompting once one or more of the above bulleted conditions are met, causing the game to end. At the end of the game, you should print the final score. You should also return an integer for which team won: return `1` if the first team won, and `2` if the second team won.

You may assume that the user enters valid non-negative integers whenever prompted, and you may assume that the game will eventually end (it won't go on forever).

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

```
Scanner console = new Scanner(System.in);
```

Call	*// typical game* `baseball(console);`	*// 10 run limit; game called* `baseball(console);`	*// extra innings* `baseball(console);`

Example Output	`Inning #1: 1 1`
	`Inning #2: 0 0`
	`Inning #3: 2 1`
	`Inning #4: 0 2`
	`Inning #5: 1 0`
	`Inning #6: 1 1`
	`Inning #7: 3 1`
	`Inning #8: 0 0`
	`Inning #9: 1 0`
	`Final score: 9 - 6`
		`Inning #1: 0 1`
		`Inning #2: 1 2`
		`Inning #3: 0 3`
		`Inning #4: 1 1`
		`Inning #5: 0 4`
		`Inning #6: 1 2`
		`Final score: 3 - 13`
			`Inning #1: 0 1`
			`Inning #2: 1 0`
			`Inning #3: 0 3`
			`Inning #4: 1 0`
			`Inning #5: 2 0`
			`Inning #6: 1 2`
			`Inning #7: 1 0`
			`Inning #8: 0 0`
			`Inning #9: 1 1`
			`Inning #10: 0 0`
			`Inning #11: 0 1`
			`Final score: 7 - 8`

Returns	`1`	`2`	`2`

(More writing space can be found on the next page.)

8. Programming (writing space)