CSE 142, Summer 2008
Midterm Exam, Monday, July 28, 2008

Name: _____

Section: _____ **TA:** _____

Student ID #: _____

Rules:

- You have 60 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>only abbreviations that *are* allowed</u> for this exam are
  `S.o.p` for `System.out.print` and `S.o.pln` 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		15
3	While Loop Simulation		15
4	Assertions		15
5	Programming		15
6	Programming		15
7	Programming		10
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
`5 * 2 * 4 - 3 * 3`	_____
`29 / 4 / 2.0 + 18 / 5 + 1.5`	_____
`30 % (4 + 3) + 16 % 20`	_____
`!(23 + 2 * 2 <= 27) && 5 % 2 == 1`	_____
`1 + 1 + "(8 - 2)" + (8 - 2) + 1 + 1`	_____

2. Parameter Mystery (15 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 arithmetic, the output does not necessarily follow the rules of addition.)

```
public class ParameterMystery {
    public static void main(String[] args) {
        int one = 4;
        int two = 3;
        int three = 10;
        int num = 17;
        int four = 3;

        racket(one, two, three);
        racket(three, four, 5);
        racket(2, two * 2, num);
        racket(num, three * one, four);
        racket(three - four, one, two);
    }

    public static void racket(int two, int one, int three) {
        System.out.println(three + " is roughly " + two + " plus " + one);
    }
}
```

3. While Loop Simulation (15 points)

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 s = 0;

     while (x > 0 && 2 * y >= x) {
         System.out.print(s + " ");
         y = y - x;
         x--;
         s = s + x;
     }
     System.out.println(s);
}
```

Method Call	Output
`whileMystery(-2, -6);`	_____
`whileMystery(2, 3);`	_____
`whileMystery(4, 8);`	_____
`whileMystery(5, 40);`	_____
`whileMystery(10, 31);`	_____

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 mystery(int a) {
    int b = 0;
    int c = 0;

    // Point A
    while (a != 0) {
        // Point B
        c = a % 10;

        if (c % 2 == 0) {
            b++;
        } else {
            b = 0;
            // Point C
        }

       a = a / 10;
       // Point D
   }

   // Point E
   return b;
}
```

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

	`a != 0`	`c % 2 == 0`	`b > 0`
Point A			
Point B			
Point C			
Point D			
Point E			

5. Programming (15 points)

Write a static method named **`graduation`** that takes a student's GPA, total credit count, and honors credit count as parameters, and returns a `String` representing that student's graduation status. The total credit count already includes the honors credits. The graduation status to return is determined by the following rules:

- Students must have completed at least 180 credits with a GPA of at least 2.0 to graduate. A student who does not meet both of these constraints should receive a return value of `"not graduating"`.
- Students who do have enough credits to graduate and sufficiently high GPAs will receive one of four return values depending on the GPA and number of honors credits:
  - All students with GPAs between 2.0 and 3.6 receive a return value of `"graduating"`.
  - Students with fewer than 15 honors credits receive a return of `"cum laude"` if their GPA is at least 3.6 but less than 3.8, and a return of `"magna cum laude"` if their GPA is at least 3.8.
  - Students with 15 or more honors credits receive a return of `"magna cum laude"` if their GPA is at least 3.6 but less than 3.8, and a return of `"summa cum laude"` if their GPA is at least 3.8.

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

Call	Value Returned
`graduation(3.87, 178, 16)`	`"not graduating"`
`graduation(1.5, 199, 30)`	`"not graduating"`
`graduation(2.7, 380, 50)`	`"graduating"`
`graduation(3.6, 180, 14)`	`"cum laude"`
`graduation(3.62, 200, 20)`	`"magna cum laude"`
`graduation(3.93, 185, 0)`	`"magna cum laude"`
`graduation(3.85, 190, 15)`	`"summa cum laude"`

You may assume that the GPA will be between 0.0 and 4.0 and that both credit counts will be non-negative integers.

6. Programming (15 points)

Write a static method named **`cheerleader`** that accepts two integer parameters *lines* and *cheers* and prints a series of "cheer" lines at increasing levels of indentation. The first parameter represents the number of lines of output to print, and the second represents the number of "cheers" per line. For example, the call of `cheerleader(2, 4)` means that you should print 2 lines of output, each containing 4 "cheers." A "cheer" is an occurrence of the word "Go" in the output. Neighboring cheers are separated by the word "Team", so 1 cheer is printed as "Go", 2 cheers as "Go Team Go", 3 cheers are printed as "Go Team Go Team Go", and so on.

The lines you print should be displayed at increasing levels of indentation. The first line displayed should have no indentation, but each following line should be intended by 3 spaces more than the one before it. In other words, the 2nd line of output should be indented by 3 spaces, the 3rd line by 6 spaces, and so on.

You may assume that both parameters passed your method will have values of at least 1.
The following calls demonstrate your method's behavior. Your method should match this output format exactly:

Call	`cheerleader(2, 1);`	`cheerleader(4, 3);`	`cheerleader(2, 4);`

Output	`Go`
	`   Go`
		`Go Team Go Team Go`
		`   Go Team Go Team Go`
		`      Go Team Go Team Go`
		`         Go Team Go Team Go`
			`Go Team Go Team Go Team Go`
			`   Go Team Go Team Go Team Go`

7. Programming (10 points)

Write a static method named **`randomRects`** that calculates and displays the area of randomly-generated rectangles. The width and height of each rectangle should be a randomly generated integer between 1 and 10 inclusive. Your method should keep generating random rectangles until an increasing sequence of four areas is printed. In other words, if the last four rectangles generated have areas of *a1*, *a2*, *a3* and *a4* such that *a1* < *a2* < *a3* < *a4*, the method should print the final message and stop. So your method will generate at least 4 total rectangles but possibly many more, stopping only when it sees 4 in a row with areas in increasing order.

The following calls demonstrate your method's behavior. Your method should match this output format exactly:

Call

```
randomRects();
```

Output

```
w: 5, h: 6, area: 30
w: 10, h: 5, area: 50
w: 2, h: 8, area: 16
w: 4, h: 4, area: 16
w: 2, h: 9, area: 18
w: 8, h: 3, area: 24
w: 7, h: 2, area: 14
w: 3, h: 10, area: 30
w: 7, h: 9, area: 63
w: 9, h: 8, area: 72
Four rectangles of increasing area.
```

Call

```
randomRects();
```

Output

```
w: 5, h: 2, area: 10
w: 6, h: 5, area: 30
w: 7, h: 6, area: 42
w: 8, h: 10, area: 80
Four rectangles of increasing area.
```

X. Extra Credit (+1 point)

Describe one of your instructors or TAs in terms of a programming construct we have learned about (`while` loops, `if` statements, etc). For example, you might write "*Jennifer is like a `Random` object because her hair sticks out in unpredictable ways.*"

(Any phrase you write that includes a programming construct and one of the TAs or instructors will get the +1 point.)