**CSE 142, Spring 2010**
**Midterm Exam, Friday, May 7, 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 ... The *only* allowed abbreviations 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 | | 15 |
| 2 | Parameter Mystery | | 15 |
| 3 | If/Else Simulation | | 15 |
| 4 | Assertions | | 15 |
| 5 | Programming | | 15 |
| 6 | Programming | | 15 |
| 7 | Programming | | 10 |
| X | Extra Credit | | +1 |
| TOTAL | **Total Points** | | **100** |

**Extra Credit (+1 point):** What score do you think you will earn on this exam? _____
*(You will get +1 point of extra credit for making any reasonable guess. Please take the question seriously.)*

## 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 |
|---|---|
| `8 - 2 * 3 + (5 - 1)` | _____ |
| `10 - 2 + "6" + 3 * 25 + (33 - 3) + 9` | _____ |
| `19 % 8 == 3 && 2 == 7 / 3` | _____ |
| `1.0 / 2 + (4.5 - 1.5) - 7 / 2` | _____ |

## 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 butters = "kenny";
        String friends = "butters";
        String kenny = "friends";
        String kyle = "stan";
        String ike = "cartman";

        stotch(kenny, friends, butters);
        stotch(ike, friends, kyle);
        stotch(kyle, ike, "ike");
        stotch(friends, "kyle", kenny);
    }

    public static void stotch(String kenny, String butters, String friends) {
        System.out.println(butters + " is " + friends + " with " + kenny);
    }
}
```

## 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 % 2 != 0) {
        a = a * 2;
    }
    if (a > 10) {
        b++;
    } else if (a < 10) {
        a--;
        b--;
    }

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

| Method Call | Output |
|---|---|
| `ifElseMystery(12, 12);` | _____ |
| `ifElseMystery(7, 4);` | _____ |
| `ifElseMystery(5, 8);` | _____ |
| `ifElseMystery(3, 42);` | _____ |

## 4. 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 funky(int a, int digit) {
    int count = 0;

    // Point A
    while (a != 0) {
        // Point B
        if (a % 10 == digit) {
            count++;
            // Point C
        } else if (count > 0) {
            count--;
            // Point D
        }
        a = a / 10;
    }

    // Point E
    return count;
}
```

| | a == 0 | a % 10 == digit | count > 0 |
|---|---|---|---|
| **Point A** | | | |
| **Point B** | | | |
| **Point C** | | | |
| **Point D** | | | |
| **Point E** | | | |

## 5. Programming

**Write a method `dominant`** that accepts three integers as parameters and returns `true` if any one of the three integers is **larger than the sum of the other two integers**. The integers might be passed in any order, so the largest value could be any of the three. If no value is larger than the sum of the other two, your method should return `false`.

For example, the call of `dominant(4, 9, 2)` would return `true` because 9 is larger than 4 + 2. The call of `dominant(5, 3, 7)` would return `false` because none of those three numbers is larger than the sum of the others. You may assume that none of the numbers is negative.

## 6. Programming

**Write a method named `invest`** that computes the interest earned on a financial investment. Once per year, the bank pays interest to the investor at a given fixed rate. As the investment earns interest, that interest is added to the investment and therefore the interest begins earning interest of its own ("compounding").

Your method should accept **three parameters**: the initial amount of the investment in dollars (as a real number such as `1500.0` for $1,500.00), the yearly interest rate (as a real number such as `3.5` for 3.5% interest) and the number of years for which to invest (as an integer such as `6` for 6 years).

Your method should print the value of the investment after each year, as well as the total amount of interest earned over all years. For example, an investment of $100.00 at 10% interest for 5 years would lead to this call:

```
invest(100.00, 10.0, 5);        // $100.00 at 10% interest for 5 years
```

The call would produce the following console output. Dollar amounts print with **2 digits after the decimal**.

```
After year 1: $110.00
After year 2: $121.00
After year 3: $133.10
After year 4: $146.41
After year 5: $161.05
Total interest earned: $61.05
```

Notice that we are not simply adding 10% of $100.00 (or $10.00) each year; that would lead to $50 of total interest earned. The first year adds 10% of $100.00, but the second year adds 10% of $110.00, or $11.00, leading to a total of $121.00. The third year adds 10% of $121.00, or $12.10, leading to a total of $133.10. The interest is **cumulative**.

You may assume that all parameter values passed are non-negative.

## 7. Programming

**Write a method named `lucky`** that accepts an integer parameter *min* and rolls a 6-sided die until it gets **four consecutive rolls in a row** that have values of *min* or less. As the method rolls the die it should print each value rolled, and then it should print a message at the end to indicate how many rolls were made. For example, the call of `lucky(3);` should print output such as the following (though the output would be different on every call because of randomness). Notice that the method stops after rolling 3, 2, 1, and 2 consecutively because these are all values of 3 or less:

```
5 2 4 6 1 3 5 5 3 2 1 2
Finished after 12 rolls.
```

A call of `lucky(5);` should print output such as the following. Notice that it continues rolling until it rolls four consecutive dice rolls produce a value of 5 or less. In this example, those values are 3, 5, 5, and 4.

```
1 3 6 3 5 5 4
Finished after 7 rolls.
```

You may assume that the parameter value passed will be between 1 and 6 inclusive.