Name:

Fundamentals

Fill in your answers in the space provided. You can check them with DrJava where possible.

---

1. The Mod Operator: %

1. What are the values of these expressions:

```
a) 10 % 0
b) 10 % 2
c) 10 % 3
d) 10 % 6
e) 10 % 20
f) 10 % 25
```

2. Given an integer x, if (x % 2) equals 0, what can we say about x?

3. Given an integer x, if (x % 2) equals 1, what can we say about x?

4. Given an integer x, can (x % 2) equal anything other than 0 or 1?

5. Given integers x and y, if (x % y) equals 0, what can we say about the relationship between x and y?

6. Given integers x and y, if x equals y, what is the value of (x % y)?

7. Given integers x and y, if x is less than y, what is the value of (x % y)?

8. What are two different real-world applications where a mod operator would be useful?

---

2. Operator Precedence

Programming language developers strive to make the languages they create unambiguous so that no matter what machine a program is run on it will run the same way. Although this is hard to accomplish in every scenario, one area in which this can be accomplished is in evaluating expressions. The rules for evaluating expressions look a first glance like what you saw in high school algebra (multiplication and division have a higher precedence than addition and subtraction, and they all associate from left to right). Since most programming languages have many operators, the rules for the order of evaluation are summarized in an "operator precedence table".

Use as a reference the **Operator Precedence Table on Resources** page of the course website (Programming section) to answer the questions. NOTE: Make sure you understand how to read the operator precedence table - higher to lower precedence *and* associativity by reading the Rules. Note: Programmers are encouraged to use parentheses for clarity.

For each expression listed below, add parentheses to show how Java will evaluate it. If you're not sure, devise experiments to try with DrJava to find out.

```
// #1
2 + 5 * 10    // Answer, since * has higher precedence than +: 2 + (5 * 10)


// #2
3 - 2 + 1


// #3
3 - 2 + 5 * 10 - 9


// #4
a < b && b < c || c < d


// #5
w = x = y = z;
```

---

3. Truth Table

Fill in the following "truth table", where T means true and F means false:

a	b	a && b	a || b	! (a && b)	(! a) && b
T	T				
T	F				
F	T				
F	F				

Use DrJava to check your answers. For example:

```
> boolean a = true;
> boolean b = true;
> a && b
true
```

---

4. Conditionals

1. Write Java code for this specification ("spec") i.e. write instructions(statements) in java syntax such that when the instructions are executed, the behavior described is achieved. Assume you are writing these statements in Dr Java Interactions pane.
   a. Assume there's an integer variable called x. If x is

```
- negative: change x to its absolute value (e.g. -5 to 5)
- in the range of 1 to 10: double x e.g. if x = 5 then x becomes 10
- greater than 10: print its value, double it, print its new value.
   Hint: use print command.
- for any other cases, do nothing
```

   b. Is it possible to do the exercise above either with

```
- 3 or more separate if statements
- an "if" statement with cascading "else if"s

     If you answer above is yes, which way is more efficient and why?
```

2. Write Java code for the specification ("spec") below:

   Using variable *year* (of type integer) to represent a calender year, print whether the year is a leap year. Note: A leap year :

   - Is a year divisible by 4
   - But not by 100 unless it’s also divisible by 400

     Example: if year = 2008, then output will be **2008 is a leap year** (again make use of print command to acheive the output statement)

Note: Go through Exercise 2 on Practice Exercise page on course site (under Programming section) for more practice.

5. Programming Structure

Complete Exercise 1 on course website under Practice Exercise section (no work needs to be handed in for this question)