Answers to Parameter Problem Set

1. True/False questions

1a)	The value of a local variable named **`i`** has no direct relationship with that of a variable named **`i`** in its caller.	T
1b)	The value of a parameter named **`x`** has no direct relationship with that of a variable named **`x`** in its caller.	T
1c)	Assigning a new value to a parameter changes the value of the corresponding variable in the caller's argument list.	F
1d)	Local variables of type **`int`** are automatically initialized to 0 when a method is called.	F
1e)	The **`return`** statement allows methods to return more than one value.	F
1f)	Predicate methods always return a value of the primitive type **`boolean`**.	T

2. Short answer

Beginning students of programming may find it hard to understand the relationships among variables in the different methods that make up a program. The instructor may hear that it would be easier if a particular variable name, such as **`total`** or **`i`**, always contained the same value in any part of the program in which that variable appeared. In your own words, describe what would be wrong with that approach?

Answer to question 2

The tremendous advantage of keeping the local variable names for each method separate is that doing so means that you can understand exactly where those variables are used without having to look at the program as a whole. A sophisticated commercial program may have hundreds of thousands or even millions of lines. If the value of a variable could be changed by any part of a large program, it would be impossible to understand all of the potential interactions in the program. Having separate name spaces for local variables also makes it much easier for teams of developers to work on a single program together. If the names are separate, I can write my methods using names I choose and not worry about whether you might use the same variable names in the methods you create.

3. Short answer

Suppose that you wanted to explain the concept of arguments and parameters to a sibling (or, more likely, a parent) with no background in programming at all. What everyday, real-world example would you use to make it clear why a procedure (or any generalized notion of a set of steps for carrying out a particular task) might want to include some form of parameterization?

Answer to question 3

There are, of course, many examples. Here, for example, are several operations and some of the parameters they might take:

- Order from a catalog: item number, quantity, size, color.
- Set up to record programs on a VCR: channel, start time, duration.
- Compute the price for sending a package: weight, destination ZIP code.

4. Short answer

In a few sentences, describe the difference between local variables and instance variables. Make sure that your answer includes any differences in (a) how those variables are declared and (b) how long the values of those variables persist.

Answer to question 4

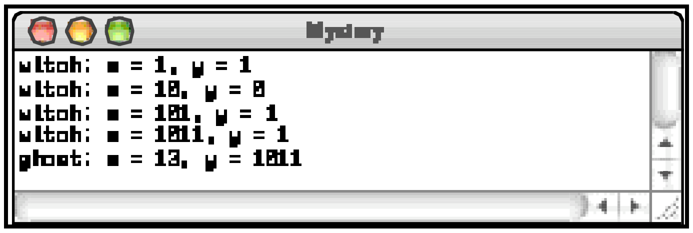
Local variables are used to keep track of values that are needed only during the evaluation of a method. They are declared inside the body of the method and may be used only in the rest of the block in which that declaration appears. All local variables are reclaimed when the method returns.

Instance variables are used to keep track of the state of an object during its lifetime and are essential when more than one method needs to have access to a value or when the state of a variable needs to be preserved between calls. Instance variable declarations appear inside the definition of a class, but outside of any method body. Instance variables exist from the time the object is created and persist as long as the object remains accessible.

5. Tracing method execution

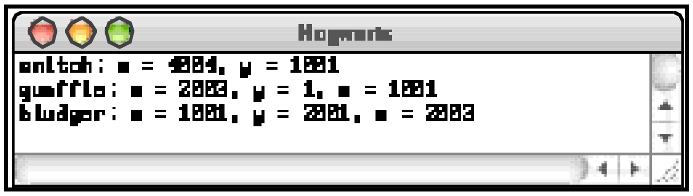
The output when each of the two programs runs is as follows.

5a)



```
witch: x = 1, y = 1
witch: x = 10, y = 0
witch: x = 101, y = 1
witch: x = 1011, y = 1
ghost: x = 13, y = 1011
```

5b)



```
snitch: x = 4004, y = 1001
quaffle: x = 2003, y = 1, z = 1001
bludger: x = 1001, y = 2001, z = 2003
```