Programming Notes 4.5: A Couple Small Things

1. Random Numbers

While writing a function to generate random numbers is extremely complicated, there is a simple way to get random numbers from a couple of library files. These files are "cstdlib" and "ctime." If you are going to use random numbers in your program, you first need to seed the random number generator with the current time. "Seeding" the random number generator means giving it a value to start with, so it can generate other random numbers from the seed. This is necessary because you will actually be using a pseudo-random number generator, and if you give it the same (or no) seed every time, it will give the same stream of random numbers every time. Therefore, you must seed it with something that is constantly changing, like the time. To do this, put the following statement at the start of your program (this is why you need "ctime"):

```
srand(time(NULL));
```

Then, to use the random number generator, you can call the function rand(). It returns a random integer value, which is not very useful, as it can be extremely massive. Hence, you can add a modulo statement after the function to convert the large value to a value between the numbers 0 and one minus whatever you modded by. Examples:

```
rand();                 -Gives random integer

rand() % 10;            -Gives random integer between 0 and 9

(rand() % 10) + 1;      -Gives random integer between 1 and 10
```

2. Increment/decrement

Instead of doing something like this:

```
variable = variable + 1;

variable = variable – 1;
```

Or even something like this:

```
variable += 1;

variable -= 1;
```

You can do this:

```
variable++;

variable--;
```

3. Indentation

Not sure if this was covered before (or if it was obvious), but you should be starting a new level of indentation for each code block—i.e. after each new set of curly brackets, or after a control flow statement, etc.

4. Microprograms

If you ever don't entirely understand something, it's always very helpful to write what I call a "microprogram," which is just a tiny program that uses whatever aspect of programming you're trying to understand. For example, you could write a microprogram that simply uses a "for" loop, and through that you can test out what you can do with a for loop, and assure that you know how to use them.

5. Default parameters

Finally, when you define parameters for your function in the function prototype, you can set one or more equal to a value. This will make it a default parameter, which means that it does not have to be passed to the function when it is called. If still can be passed like any other parameter, but if it's not, it will simply have the default value you defined for it. Remember that all default parameters have to come after the required parameters, because if they are in the middle, the complier will not know if you want to pass the required or default parameter. Example:

```
int function(int, int, int = 2);

int function(int x, int y, int var) {

  return (x + y) / var;

}

int int1 = function(3,7);          -This is valid, and will return the value 2

int int3 = function(3,7,2);        -This is also valid, and will return the value 5
```