Module B.2: More Arduino Projects

**2. Locate on-line documentation that describes the C language “for” loop.**

1. **What is the index and how is it used?**

After creating a setup() function, which initializes and sets the initial values, the loop() function does  what its name suggests, and loops consecutively, allowing your program to change and respond. Use it to actively control the Arduino board.

**b. When does the for loop end?**

The for loop will end based on the  index and base **(e.g. i=0; 1<10, 1++).** In this example the program will loop 10 times and end

**c.  How is a “for” loop different from a “while” and a “do” loop?**

**While loops:** will loop continuously, and infinitely, until the expression inside the parenthesis, () becomes false. Something must change the tested variable, or the **while** loop will never exit. This could be in your code, such as an incremented variable, or an external condition, such as testing a sensor.

**3.  Research the “<” Comparitor.**

**a.    List all the other comparitors defined for the C language.**

**x == y** (x is equal to y)

**x != y** (x is not equal to y)

**x < y** (x is less than y)

**x > y** (x is greater than y)

**x <= y** (x is less than or equal to y)

**x >= y** (x is greater than or equal to y)

**b.    Modify the “for” loop to use the “<=” comparitor**

**Before**: for (int i = BASE; i < BASE + NUM; i ++)

**New** :for (int i = BASE; i =+ BASE + NUM; i ++)

4. **Research the “++” incrementor operator.**

**a.    Explain how this is different from the “=+ 1” assignment**

var x = 0; x++;          // x is now 1  
var y = 0; y = y + 1;    // y is now 1

However, in JavaScript every assignment has a value**, and it's always the value that b has:**var value = (x = 99);  
console.log( value );     // 99  
**And that is useful  because you can use it to set many variables to the same value:**

a = b = c = d = 1;  
 **But there is one important difference between the two notations in question:**statement              its value  
––––––––––––––––––––––––––––––––  
x = x + 1                  x + 1  
x++                        x        (oops!)  
So although x++ does increase the value of x by 1, the value of the expression "**x++**" itself remains x, so if you then go on and write that value back into x (which you do by saying **x = x++)**, **you've just set x back to its old value:**var x = 99;  
x = x++;  
console.log( x );     // 99  
**So for your loop to work you mustn't do that with your loop counter.**If you really want to, you can do this instead: **x = ++x; –** that'll work fine because, unlike with **x++**, the value of **++x** equals x+1. But – don't do that! It'll just confuse anyone trying to make sense of your code (for instance, yourself). Having only **x++** is cleaner and simpler.

**b.    Modify the “for” loop to use the “=+” assignment**

**Before:** for (int i = BASE; i < BASE + NUM; i ++)

**New** :for (int i = BASE; i =+ BASE + NUM; i ++)