**Review Questions:** What do the following program segments do:

double x = 4.0;
double y = 12.0;
int i = 3;
**i++;**double z = (x+y)/i;
LCD.drawString(“z=” + z, 1, 1);

Note that the ++ operator “increments by 1” the integer variable to which it is applied. So, if x is an integer variable with value 6 and you execute the command x++ then the value of x after that execution will be 7.

Thus, the above output will be “z = 4.0”

UltrasonicSensor sonar = new UltrasonicSensor(SensorPort.S1);
if (sonar.getDistance() >= 30)
{
 robot.forward();
}
else
{
 robot.backwards();
}

If the distance sensor is “far away” from an obstacle (to be precice: 30 or more cm away), then the robot will drive forward. Otherwise, i.e. if it is closer than 30 cm, the robot will drive backwards. This decision is made only once!

TouchSensor touch = new TouchSensor(SensorPort.S1);
LightSensor light = new LightSensor(SensorPort.S2);
while ((touch.isPressed()) && (light.readValue() < 35))
{
 robot.forward();
}
robot.stop();

Our first example of a loop: Note that the && means “and” (and || means “or” and ! means “not””). So, this code means that as long as the touch sensor is pressed and the light sensor registers relative darkness (less than 35), the robot will continue to drive forward. As soon as either the touch sensor is no longer pressed or the light intensity goes above (or both), robot will stop.

int count = 2;
while (count <= 10)
{
 LCD.drawString(“Count = “ + count, 1, 1);
 count = count + 2;
}
LCD.drawString(“Count = “ + count, 1, 1);

This is a typical example of a “counting” loop. The loop, basically, counts up from 2 to 10 by 2’s, then exits. So, it will print out:

Count = 2
Count = 4
Count = 6
Count = 8
Count = 10

Then the loop is over but because of the statement after the loop it will print out one more “Count = 12”.

double x = 80.0
while (x != 1)
{
 x = x / 2.0;
 LCD.drawString(“x = “ + x, 1, 1);
}

A variation of the counting loop from above. Each time the loop executes, will divide the value of x by two and reassign the answer back to x. This will continue as long as x is not 1.

WARNING: This will result in an infinite loop!!!!!!

First, x = 80. Then x becomes 40, then 20, then 10.0, then 5.0, then – because the variable is of type double (i.e. decimal) 2.5, then 1.25, then 0.625, etc. In particular, it will never be equal to 1 so the loop will never end!!! This is bad! As a rule of thumb:

 NEVER USE AND EQUAL (OR NOT EQUAL) FOR DOUBLE VARIABLES

To fix it, there are two solutions. The obvious one would be to replace the test for the loop by (x >= 1). That would do the trick and the last number printed out would be 1.25.

Another solution, less obvious, would be to make the variable x an “int”, not a double. Because then the division will always return an integer, so the value of x would be 80, 40, 20, 10, 5, 2, and 1.