LARSON—MATH 255–CLASSROOM WORKSHEET 41
Problems & Interacts

1. (a) Start the Chrome browser.
   (b) Go to `http://cocalc.com`
   (c) Login using **your VCU email address** .
   (d) Click on our class Project.
   (e) Click "New", then "Worksheets", then call it **c41**.
   (f) For each problem number, label it in the Sage cell where the work is. So for Problem 2, the first line of the cell should be `#Problem 2`.

More Interacts!

There is a collection of examples of Sage INTERACTS at `http://wiki.sagemath.org/interact/`. Let's look at a few of these examples to see the kinds of things you can do with Sage.

Julia Set

You can read about the Julia set at `http://en.wikipedia.org/wiki/Julia_set`. It is a common example of *complex dynamics* that can be illustrated with cool pictures.

2. First recall that the complex numbers have the form $a + bi$ where $i = \sqrt{-1}$. Let `z=5+3*i`. You can think of this as the vector which points from the origin to $(5, 3)$.

3. z has a length (or *magnitude*. Find it with `norm(z)`. Use `n(_)` to get a numerical approximation. Similarly, every point in the plane can be viewed as a complex number, and with an associated magnitude.

4. The secret to the Julia Set INTERACT is the function `complex_plot()` which associates colors to values of a complex function. Consider the function $f(z) = z$ where z is a complex number. Make a pretty plot of this using `complex_plot(f, (-5, 5), (-5, 5))`. Remember to tell Sage that z is a variable.

5. Here's the description you get if you evaluate `help(complex_plot)`: " 'complex_plot' takes a complex function of one variable, f(z). The magnitude of the output is indicated by the brightness (with zero being black and infinity being white) while the argument is represented by the hue (with red being positive real, and increasing through orange, yellow, ... as the argument increases)." Now try `complex_plot(z**2, (-5, 5), (-5, 5))`. Can you explain the lines in the picture?

6. Most of the following code is defining the input numbers and various tricks for fast display of the result. The main bits of code are the definition of $f(z)$ and the call to `complex_plot`.

```
@interact
def julia_plot(expo = slider(-10,10,0.1,2),
      iterations=slider(1,100,1,30),
      c_real = slider(-2,2,0.01,0.5),
      c_imag = slider(-2,2,0.01,0.5),
      zoom_x = range_slider(-2,2,0.01,(-1.5,1.5)),
      zoom_y = range_slider(-2,2,0.01,(-1.5,1.5))):
    var('z')
    I = CDF.gen()
    f(z) = z^expo + c_real + c_imag*I
    ff_j = fast_callable(f, vars=[z], domain=CDF)

    def julia(z):
      for i in range(iterations):
         z = ff_j(z)
         if abs(z) > 2:
            return z
      return z
    print("z <- z^{} + ({} + {})*I".format(expo, c_real, c_imag))

    complex_plot(julia, zoom_x,zoom_y, plot_points=200, dpi=150).show(frame=T
```

This last line should end with `.show(frame=True, aspect_ratio=1)` Try different input values for this set. If you think this is cool, maybe you might be interested in a course in *Discrete Dynamical Systems*!

Random Walks

7. Start at the origin on the number line. At each time step take a (random) step one unit to the right or one unit to the left. I have heard that you will (with probability 1) return to the origin at some point, Is this true? How can we investigate this experimentally?

8. If it is true, how many steps does it take on average to return to the origin?

Getting your classwork recorded

When you are done, before you leave class...

(a) Click the "Make pdf" (Adobe symbol) icon and make a pdf of this worksheet. (If CoCalc hangs, click the printer icon, then "Open", then print or make a pdf using your browser).

(b) Send me an email with an informative header like "Math 255 - c41 worksheet attached" (so that it will be properly recorded).

(c) Remember to attach today's classroom worksheet!