Last name _____

First name _____

LARSON—MATH 255–CLASSROOM WORKSHEET 11
Scatter Plots & Recursion.

1. Log in to your Sage/Cocalc account.

   (a) Start the Chrome browser.

   (b) Go to `http://cocalc.com` and sign in.

   (c) You should see an existing Project for our class. Click on that.

   (d) Click "New", call it **c11**, then click "Sage Worksheet".

   (e) For each problem number, label it in the Sage cell where the work is. So for Problem 1, the first line of the cell should be `#Problem 1`.

Recursion

A **recursive** function is a function that calls itself. It must always have a *base case* so that the recursion eventually stops.

2. The *gcd* of 2 non-negative integers is their *greatest common divisor*. The following recursive function calculates the gcd of integers a and b using the fact (which can be proved) that, if $a \geq b$ then $\gcd(a, b) = \gcd(a - b, b)$. It uses the fact that $\gcd(0, a) = \gcd(a, 0) = a$, for any non-negative integer a, as the base case.

```
def gcd(a,b):
  if a==0 or b==0:
      return max(a,b)
  else:
      return gcd(max(a,b)-min(a,b),min(a,b))
```

3. The Fibonacci sequence F_n is defined as follows $F_0 = 0$, $F_1 = 1$ and $F_n = F_{n-1} + F_{n-2}$ for $n > 1$. Write a recursive function `fib(n)` that computes the n^{th} Fibonacci number.

4. **Challenge**. Define a non-recursive (iterative) function `fib2(n)` that computes the nth Fibonacci number.

5. **Investigation Continued**. Start with any positive integer x. If x is even divide by 2. If x is odd, multiply by 3 and add 1. Repeat. Try this for several initial starting numbers x. What happens?

6. Define a function `collatz(x)` that returns x if x is one, returns `collatz(3x+1)` if x is odd, and returns `collatz(x/2)` if x is even. This will be a recursive function (since it calls itself). What is the base case?

7. Try `collatz(1)`, `collatz(2)`, `collatz(3)`, `collatz(3)`, `collatz(4)`, `collatz(5)`, `collatz(6)`, `collatz(7)`, and `collatz(100)`.

8. The output of `collatz(x)` is not very interesting. In fact, it looks like nothing is happening. Add a print statement to your code to print x at the beginning of the function. (This makes your code *verbose*.) Now reevaluate and rerun your code.

Random Values

9. `random()` returns a random number in $[0, 1]$. Execute it a few times to see what you get.

10. Define a function `my_mood()` which prints "I'm happy" or "I'm sad" randomly.

```
def my_mood():
    if random()<.5:
        print "I'm happy"
    else:
        print "I'm sad"
```

11. Use `random()` to define a function `coin_flip()` which randomly returns the string "H" (for heads) half the time and returns the string "T" (for tails) half the time. Try it a few times; your results will vary.

12. Make an *interactive* coin flipping program:

```
@interact
def i_coins(n=input_box(5)):
    for x in [1..n]:
        print coin_flip()
```

Try different numbers in the box.

It is often useful to generate random integers. It only makes sense to generate random integers from within some range of integers. We do this with `randint()`.

13. Evaluate `randint(5,100)` a few times; your results will vary. This will generate random integers in the range $[5, 100]$, including both endpoints.

14. Now try the following function. Evaluate it a few times; your results will vary!

```
def sybil():
    print "My favorite number is {}!".format(randint(1,50))
```

15. **Investigate**. Does `randint()` produce a *uniform distribution*? (That is, as you repeat experiments of `randint(a,b)` are the number of produced outcomes of each possible integer roughly the same? Do some experiments!)