LARSON—MATH 255–CLASSROOM WORKSHEET 11
Recursion & Random Numbers

1. (a) Start the Chrome browser.
   (b) Go to `http://cocalc.com`
   (c) You should see an existing Project for our class. Click on that.
   (d) Click "New", then "Sage Worksheet", then call it **c11**.
   (e) For each problem number, label it in theSAGE cell where the work is. So for Problem 1, the first line of the cell should be `#Problem 1`.

Follow-up

2. Write a function `to_cartesian(r,`θ`)` that takes any pair (r, θ) in polar coordinates and converts it to Cartesian coordinates (x, y).

Review

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

3. Here is an example of a recursive definition of the *factorial* function. The base case here is the case where the input is 0 or 1.

```
def facto1(n):
    if n==0 or n==1:
        return 1
    else:
        return n*facto1(n-1)
```

4. It is often intuitive to define a function recursively, but usually the same function can be defined without recursion. Here is a function `facto2(n)` that does the same thing as `factorial(x)` but is **not** recursive. Test it to make sure it gives the same results.

```
def facto2(n):
    result=1
    if n==0:
        return result
    for i in [1..n]:
        result=result*i
    return result
```

5. 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))
```

More Recursion

6. **Fibonacci!** 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 nth Fibonacci number.

7. Try this for small values of n to make sure that it works, then try it for $n = 10, 20, 30, 40, 50$. Does it finish? If not, why not?!?!

8. Write a non-recursive (iterative) `gcd` function.

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. Run your coin flipping program 100 times and collect data. A random coin flipping program should come up heads about half the time. How many times do you get heads?

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 - c11 worksheet attached” (so that it will be properly recorded).

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