CISC106 Summer 2011 Lab08

- This lab is due on Friday, August 12 at 1:00 PM EDT (right before the course final) on Sakai.

Preparation (do not submit for grading)

1. Start up the MATLAB interpreter by typing `matlab -nojvm` at the command line. This will give you an interactive interpreter much like Python's. Type some statements into the interpreter and see what happens (use the cheatsheet to help you get the right syntax.)
2. MATLAB has a nice GUI environment that you can use by giving the `matlab` command without the `-nojvm` flag. Feel free to run it and explore.

Programs (to be graded)

Below you will see implementations of various functions from earlier in the semester. You should take each of these Python implemenations and translate them to corresponding MATLAB implementations. N.B.: in MATLAB you actually call *files* rather than *functions*. That last sentence is probably confusing, so download the example `composition1.m` from the lab folder on the course website. If that file is in the current directory from which you start the MATLAB interpreter, you can call the first function in it as such: `composition1(10)`. Please note that the fact that the first function is also called `composition1` is technically irrelevant (although in general you should name the first function in a file the same name as that file to avoid confusion.)

1.
```
def box_orders(liquid_density, inventory_weight):
    return math.ceil(inventory_weight/max_contents(liquid_density))

def total_weight(volume, density):
    return volume * density

def max_contents(liquid_density):
    return total_weight(100**3, liquid_density)
```

2.
```
def sum_even(x):
    sum = 0
    for i in range(x + 1):
        if i % 2 == 0:
            sum += i

    return sum
```

3.
```
def bill_amount(data, base):
    total = min(data, 100) * base
    data = max(data - 100, 0)
    total += min(data, 400) * (base*1.33 + 0.05)
    data = max(data - 400, 0)
    total += min(data, 1000) * (base * 1.44 + 0.08)
    data = max(data - 1000, 0)
    total += data * base * 2

    return total
```

4.
```
def ackermann(m, n):
    if m == 0:
        return n + 1
    elif n == 0:
        return ackermann(m - 1, 1)
    else:
        return ackermann(m - 1, ackermann(m, n - 1))
```

You should have a bunch of .m files (four to be exact.) Submit them and any other docs required by your TA on Sakai.