# COMPUTER SCIENCE 50100
# Computing for Science and Engineering
FALL 2015

ASSIGNMENT # 2 (35 points)

September 9

**Due Wednesday, September 30 at 10:30 am** This assignment covers Section 2.3–2.6 of the class notes.

1. (8 points) The problem is to write a module `funcf.py` implementing two versions of a function `o` that takes as it arguments two functions `f` and `g`, each of which maps a `float` to a `float`. The function `o` is to return the composition `f`$\circ$`g`.

   (a) (5 points) The first implementation, call it `o1`, should make no use of the `lambda` operator.

   (b) (3 points) The second implementation, call it `o2`, should make no use the `def` command—none at all.

   Here is a template with a unit test:

```
#!/usr/bin/env python  # _funcf.py
def o1(f, g):
    ...
o2 = ...
if __name__ == '__main__':
    def f(x): return x*x + 1
    def g(y): return y*y - 2
    h = lambda z: z*z +3*z
    for o in [o1, o2]:
        fog = o(f, g)
        for x in [2, 3, 5]:
            print(fog(x) == f(g(x)))
        hogof = o(h, o(g, f))
        for x in [-1, -2]:
            print(hogof(x) == h(g(f(x))))
```

2. (10 points) sec 2.5
   Write a script that, from the current working directory and every nested subdirectory, removes every file whose name ends with "~", but *only if* there is another file in the same directory with the same name except for the tilde. Possibly useful functions: `os.getcwd`, `os.path.join`, `os.remove`, `os.walk`. Here is a testing program:

```
#!/usr/bin/env python  # cleanTest.py
import os, subprocess
os.mkdir('life')
os.chdir('life')
os.mkdir('Eukarya')
os.mkdir('Bacteria')
```

```
open('methanobacteria~', 'w').close()
open('halobacteria', 'w').close()
open('halobacteria~', 'w').close()
os.chdir('Bacteria')
open('cyanobacteria~', 'w').close()
open('spirochetes~', 'w').close()
os.chdir(os.pardir+os.sep+'Eukarya')
os.mkdir('Animals')
os.chdir('Animals')
open('pigs', 'w').close()
open('pigs~', 'w').close()
open('men', 'w').close()
open('men~', 'w').close()
os.chdir(os.pardir+os.sep+os.pardir+os.sep+os.pardir)
for label in ['Before:', 'After:']:
    print(); print(label)
    print(os.listdir('life'))
    print(os.listdir('life'+os.sep+'Bacteria'))
    print(os.listdir('life'+os.sep+'Eukarya'))
    print(os.listdir('life'+os.sep+'Eukarya'+os.sep+'Animals'))
    subprocess.call('./clean.py')
import shutil
shutil.rmtree('life')
```

3. (17 points) sec 2.6
   Construct a Python module `randNo.py` that defines a `RandNo` class. Given a file of probabilities $p_i$, $i = 1, 2, \ldots, n$, construct a function that generates $i$ with probability $p_i$.

   (a) The constructor is given a list of probabilities $p_i$, $i = 0, 2, \ldots, n-1$, where $p_i$ is the probability that an instance of a `RandNo` has the value $i$ and creates an independent "data structure" that expedites instantiation and addition. If the probabilities do not add up to 1 give or take $10^-7$, raise a `ValueError` with the format string `'Probabilities sum to %.7f.'`.

   (b) Define a call method that effects instantiation. The call should take only $\mathcal{O}(\log n)$ computing time where $n-1$ is the maximum value.

   (c) Define a `str` method for the `str` function, as illustrated below.

   (d) Define a `repr` method identical to the `str` method.

   (e) Define a method that enables two `RandNo`s to be added using the operator `+`. The result should have the correct probability distribution.

   Possibly useful functions: `random.random`. Here is a template with a unit test:

```
#!/usr/bin/env python  # _randNo.py
import sys, copy, random
class RandNo(object):
    ...
if __name__ == '__main__':
```

```
for denom in [7, 7.0001, 6.9999]:
    pr = [1/denom]*7
    try:
        RandNo(pr)
    except ValueError as e:
        print(e)
print(RandNo([0.51, .49]))
print(repr(RandNo([0.47, 0.53])))
histogram = [0]*4
N = RandNo([0.3, 0.1, 0.4, 0.2])
for k in range(1000):
    histogram[N()] += 1
print(histogram)
import time
for power in [12, 13]:
    n = 2**power
    N = RandNo([1/n]*n)
    t0 = time.time()
    for k in range(1000): N()
    t1 = time.time()
    print('for n = %d, computing time is %f' % (n, t1 - t0))
Die = RandNo([0.] + [1./6.]*6)
ThreeDice = Die + Die + Die
histogram = [0]*19
for k in range(1000):
    histogram[ThreeDice()] += 1
print(histogram[3:])
```

**Send your solutions electronically** You should submit one PDF file containing your source code for problems 1–3. Additionally, send your source code for each problem as separate files.