CISC-121 Winter 2013
Week 7 Lab Assignment: Ancient Egyptian Fractions

This is the assignment for the labs in Week 7 (February 25 – March 1). This assignment will not be handed in or graded.

Deadline:
Please complete your solution by Monday March 4. If you fall behind, it's very hard to catch up. Please don't look at my solution until you have worked through the lab and completed your own solution. In terms of learning, looking at my solution is a very poor substitute for creating your own.

Introduction:
The ancient Egyptians were capable of intricate and precise mathematical operations at a time
when much of Europe was still at the stage of counting "one, two, a lot ..." In fact, northern Africa and Arabia formed one of the world's centres for mathematical and scientific research (along with Asia) for thousands of years. As you probably know, both the words *algebra* and *algorithm* derive from the work and name of an Arabic mathematician. Despite their mathematical sophistication, the ancient Egyptians had an interesting self-imposed difficulty when it came to working with fractions: they believed that any fraction which was not the reciprocal of an integer (i.e. not of the form 1/n) was unclean or improper. Any fraction of the form x/y, where x is not a factor of y, had to be written as the sum of reciprocals. For example, 3/4 can be written as 1/2 + 1/4. Of course this is always possible, since x/y = 1/y + 1/y + 1/y + ... + 1/y where there are x terms in the sum (for example, 4/5 = 1/5 + 1/5 + 1/5 + 15). It may not be obvious that x/y can always be written as the sum of reciprocals in which all the denominators are different, but this happens to be true. (As an exercise, try to prove this.)

In the twelfth century AD an Italian mathematician named Leonardo of Pisa, whom we now remember as Fibonacci, devoted much of his life to bringing the advanced mathematics of Africa and Arabia to Europe. He was almost single-handedly responsible for introducing Europe to the decimal numerals we now use, replacing the awkward Roman Numeral system that was then in use throughout Europe. Unfairly, despite a large volume of original work and a number of extremely important translations, he is now mostly remembered for the so-called Fibonacci Sequence.

One of Fibonacci's theorems deals with the Egyptian system of writing fractions. He proved that the following algorithm:

```
To represent x/y as the sum of distinct reciprocals:
z = x/y
while z != 0
     let n be the smallest integer such that 1/n <= z
     add 1/n to the list of reciprocals
     z = z - 1/n
```

is guaranteed to produce a finite list of distinct reciprocals that sum to the original value of x/y

For example, consider x/y = 4/5

```
    z = 4/5
    1/1 > z
    1/2 <= z so
        add 1/2 to the list
        z = 4/5 - 1/2 = 3/10
    1/3 > z
    1/4 <= z so
        add 1/4 to the list
        z = 3/10 - 1/4 = 2/40
    1/5 > z
    1/6 > z
    1/7 > z
    ...
    1/19 > z
    1/20 <= z so
        add 1/20 to the list
        z = 2/40 - 1/20 = 0
    z == 0 so we stop
```

Thus 4/5 = 1/2 + 1/4 + 1/20

Assignment

Your assignment is to write a Python function that implements this algorithm recursively. In other words, you need to replace the loop in the pseudo-code above with an appropriate recursive call. Your function must take two integer parameters representing the numerator (x) and denominator (y) of the fraction, and return a list of the denominators of the reciprocals that sum together to make the fraction x/y.

Your program must ask the user to enter the numerator and the denominator (if you use easygui you will want to over-ride the default limit on the maximum value that can be entered). Your program must display the denominators of the reciprocals that make up the solution.

Note that this algorithm requires that the target number be in the range [0 .. 1], so you need to test for this before you start to solve the problem. But doing this test every time you make a recursive call is a waste of time. The solution is to “hide” your recursive function behind another function whose sole job is to check the numerator and denominator values.

This should look something like this

```
def recursive_fraction(x,y):
     # the recursive algorithm
     return answer

def fraction_as_reciprocals(x,y):
     # check to make sure parameters are ok
     # if they are ok:
          return recursive_fraction(x,y)
     # else:
          print error message
          return None
```

Now your program calls **fraction_as_reciprocals()** which checks the values and then actually initiates the call to **recursive_fraction()**.

One of the essential aspects of this algorithm is that the computations must be exact – but computers are notorious for not being able to do arithmetic on non-integers precisely. For example, try running this code:

```
x = 0
r = 1.0/3
for i in range(12):
     x += r
print x
if x == 4:
     print "addition gave correct result"
else:
     print "addition gave round-off error"

x = 4
r = 1.0/3
for i in range(12):
     x -= r
print x
if x == 0:
     print "subtraction gave correct result"
else:
     print "subtraction gave round-off error"
```

and you will see what I mean.

To deal with this, we need to do the calculations for this assignment using only integer arithmetic because there is no round-off error when computing a+b, a-b, or a*b when a and b are both integers. For example, if we want to compare the fraction
x/y
to the fraction
a/b
we can compare **x*b** to **y*a** and not have to worry about round-off error. This works because x/y < a/b is true exactly when x*b < y*a (and the same is true for testing x/y = a/b and x/y > a/b)

Similarly, to compute **x/y – a/b**, rather than perform the divisions and then the subtraction, we should use the following:
x/y – a/b = (x*b – y*a) / (y*b)
to determine the numerator (x*b – y*a) and denominator (y*b) of the result.

This means that you should each fraction as a pair of integers (numerator and denominator). You may wish to practice a bit of object-oriented programming by defining a "fraction" class.