CSE 30
Programming Abstractions: Python
Lab Assignment 3

Recall that in pa4 you wrote a function called `CF(L)` which converted a continued fraction represented by a list of integers `L`, into a Rational object. A better name for that function would probably have been `ContinuedFraction2Rational(L)`, since that was exactly the transformation it performed. Let's shorten the name to `CF2R(L)`. This new name suggests that it might have an inverse, i.e. a function taking a Rational object as input, and returning a list representing the continued fraction for that rational number as output. The obvious name for such a function is `R2CF(x)`.

How can this inversion process be accomplished? Let's begin with the example $1137/158$. We perform the (by now familiar) Euclidean algorithm for finding the GCD of the two integers $a = 1137$ and $b = 158$.

$$1137 = 7 \cdot 158 + 31 \longrightarrow \frac{1137}{158} = 7 + \frac{31}{158}$$

$$158 = 5 \cdot 31 + 3 \longrightarrow \frac{158}{31} = 5 + \frac{3}{31}$$

$$31 = 10 \cdot 3 + 1 \longrightarrow \frac{31}{3} = 10 + \frac{1}{3}$$

$$3 = 3 \cdot 1 + 0 \longrightarrow \frac{3}{1} = 3 + 0$$

Upon dividing each equation by its respective divisor, we obtain a list of equations with which we can build up the continued fraction for $1137/158$.

$$\frac{1137}{158} = 7 + \frac{31}{158} \qquad = 7 + \frac{1}{158/31}$$

$$= 7 + \frac{1}{5 + \frac{3}{31}} \qquad = 7 + \frac{1}{5 + \frac{1}{31/3}}$$

$$= 7 + \frac{1}{5 + \frac{1}{10 + \frac{1}{3}}}$$

$$= \ [7, 5, 10, 3]$$

Thus, the list representation of the continued fraction for a given rational number is nothing more than the list of quotients in Euclid's algorithm for the GCD of the numerator and denominator.

Before we go any further, we should ask whether there is a unique continued fraction for every rational number. Without this property, there would be no inverse function for `CF2R()`. Unfortunately, the answer is no, as the following examples show. We have

$$\frac{151}{21} = 7 + \frac{1}{5 + \frac{1}{4}} \qquad = [7, 5, 4]$$

and

$$\frac{151}{21} = 7 + \frac{1}{5 + \frac{1}{3 + \frac{1}{1}}} \quad = [7, 5, 3, 1].$$

One checks that $[x_0, x_1, \ldots, x_{n-1}, 1] = [x_0, x_1, \ldots, x_{n-1} + 1]$ for any integers $x_0, x_1, \ldots, x_{n-1}$. Another kind of redundancy occurs when some of the integers in the list are negative. For instance

$$\frac{18}{11} = 2 + \frac{1}{-3 + \frac{1}{4}} \qquad = [2, -3, 4]$$

and

$$\frac{18}{11} = 1 + \frac{1}{1 + \frac{1}{1 + \frac{1}{1 + \frac{1}{3}}}} = [1, 1, 1, 1, 3].$$

Finally, one checks that $[a, 0, b] = [a + b] = a + b$ for any integers a and b. Fortunately, these are the only redundancies possible. Therefore, if we require that our list representation contains only positive integers, and does not end in 1, then the continued fraction for a given (positive) rational number is unique, and is given by the list of quotients in the Euclidean algorithm, as illustrated above.

To start this project, create a module called MoreContinuedFractions.py, and import your module rational.py, as you did in pa4. Rename your function `CF(L)` from that project to `CF2R(L)`, then write its inverse `R2CF(x)`, which takes a Rational object and returns a list, as described above. Check that both of the expressions `CF2R(R2CF(x))==x` and `R2CF(CF2R(L))==L` return True for any Rational object `x` and any list `L`.

The expressions we've studied so far are sometimes called *regular*, *simple* or *ordinary* continued fractions, to distinguish them from *generalized continued fractions*, which are expressions of the form

$$x = b_0 + \cfrac{a_1}{b_1 + \cfrac{a_2}{b_2 + \cfrac{a_3}{b_3 + \ddots \; + \cfrac{a_n}{b_n}}}}$$

Written this way, generalized continued fractions are evidently encoded using two lists. The numerators $[a_1, a_2, a_3, \ldots, a_n]$, and the (partial) denominators $[b_0, b_1, b_2, b_3, \ldots, b_n]$, whose lengths differ by 1. We can turn these two lists into a single list by just interleaving them, obtaining $[b_0, a_1, b_1, a_2, b_2, a_3, b_3, \ldots, a_n, b_n]$. This composite list, which is necessarily of odd length, can be expressed uniformly as

$$[x_0, x_1, x_2, x_3, x_4, \ldots, x_{n-1}, x_n] = x_0 + \cfrac{x_1}{x_2 + \cfrac{x_3}{x_4 + \ddots + \cfrac{x_{n-1}}{x_n}}}$$

Observe that n must be even, so that the list length $n+1$ is odd. With this convention, the list elements having odd indices are the numerators, and elements with even indices are the (partial) denominators. A few examples are now in order.

$$[1,2,3,4,5] = 1 + \cfrac{2}{3 + \cfrac{4}{5}} = 1 + \frac{2}{19/5} = 1 + \frac{10}{19} = \frac{29}{19}$$

$$[0,4,1,1,3,4,5,9,7] = 0 + \cfrac{4}{1 + \cfrac{1}{3 + \cfrac{4}{5 + \cfrac{9}{7}}}} = \cdots \text{exercise} \cdots = \frac{160}{51}$$

Like ordinary continued fractions, generalized continued fractions can be defined recursively.

$$[x_0, x_1, x_2, x_3, x_4, \ldots, x_{n-1}, x_n] = x_0 + \cfrac{x_1}{\left(x_2 + \cfrac{x_3}{x_4 + \ddots + \cfrac{x_{n-1}}{x_n}} \right)}$$

$$= x_0 + \frac{x_1}{[x_2, x_3, \ldots, x_{n-1}, x_n]}$$

Your next step in this assignment will be to write a recursive function called `GCF2R(L)` that takes a list L (of odd length) representing a generalized continued fraction, and returns the corresponding rational number as an object of type Rational. What if `GCF2R()` is called on an even length list? A simple way to avoid raising an exception is by adding a base case for lists of length 0, which just returns Rational(1). This effectively appends 1 to the list, making it even length.

The Wikipedia article https://en.wikipedia.org/wiki/Generalized_continued_fraction#%CF%80 contains a number of (infinite) generalized continued fractions for π. One of the best is

$$\pi = 0 + \cfrac{4}{1 + \cfrac{1^2}{3 + \cfrac{2^2}{5 + \cfrac{3^2}{7 + \ddots}}}} = [0, 4, 1, 1^2, 3, 2^2, 5, 3^2, 7, 4^2, 9, \ldots\ldots]$$

By truncating this expression at any point, we can obtain a good rational approximation to π. For the final part of this assignment, write a function called `pi_gen()` returning a generator object that produces the sequence $[0, 4, 1, 1^2, 3, 2^2, 5, 3^2, 7, 4^2 \ldots\ldots] = [0, 4] + [2k-1, k^2]_{k=1}^{\infty}$. Write a `main()` function in your

module MoreContinuedFractions.py that uses all of this infrastructure to determine a rational approximation to π. Use enough terms in the above sequence so that, when converted to a Decimal object, your approximation is correct to 100 decimal places. (Use Google to find the first 100 decimal digits of π and check your answer.) Finding the necessary number of terms from the above sequence for this approximation will take some trial and error.

Once this is done, use your rational approximation to π, and your function `R2CF()` to obtain the equivalent ordinary continued fraction, as a list. Print the following to the standard output stream.

a blank line
your rational approximation to π
a blank line
the equivalent Decimal object, correct to 100 places
a blank line
the equivalent ordinary continued fraction
a blank line

Test your functions: `CF2R()`, `R2CF()`, `GCF2R`, and `pi_gen()` independently to make sure they meet specifications, since this is what the graders will do. Submit **both** files

MoreContinuedFractions.py (new file for this project)
rational.py (unchanged from pa4)

to Gradescope before the due date.