CSCI 301, Lab # 2

Winter 2017

Due: Your program, named `lab02.rkt`, must be submitted to Canvas before midnight, Tuesday, January 24.

Background: An infinite *continued fraction* is an expression of the form

$$f = \cfrac{n_1}{d_1 + \cfrac{n_2}{d_2 + \cfrac{n_3}{d_3 + \ddots}}}$$

Where n_i and d_i are functions of the index, i. For example, if $n_1 = 1$ and $d_1 = 1$ for all i, then we get an expression equal to the reciprocal of the golden ratio:

$$\phi^{-1} = \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{1 + \ddots}}}$$

One way to approximate an infinite continued fraction is to truncate the expansion after a given number of terms, giving a *k-term finite continued fraction*, which has the form:

$$f = \cfrac{n_1}{d_1 + \cfrac{n_2}{d_2 + \cfrac{n_3}{\ddots + \cfrac{n_k}{d_k}}}}$$

Programming:

Suppose that `n` and `d` are procedures of one argument (the term index i) that return the n_i and d_i of the terms of the continued fraction. Define a procedure `cont-frac` such that evaluating `(cont-frac n d k)` computes the value of the k-term finite continued fraction.

Now test your continued fraction implementation with several famous continued fractions:

1. Find ϕ, which is approximately 1.61803398875, but you don't have to find that many digits.

   The main function call for this could look like:

   ```
   (cont-frac (lambda (x) 1.0)
              (lambda (x) 1.0)
              100)
   ```

   Then take the reciprocal and compare it to the golden ratio.

2. In 1737 Leonhard Euler showed that $e - 2$ was equal to the continued fraction where all the numerators, n_i were 1 and the denominators, d_i were $1, 2, 1, 1, 4, 1, 1, 6, 1, 1, 8, \ldots$ Check your continued fraction procedure by using it to approximate e. You can find an accurate value of e in Scheme with `(exp 1)`.

   The main function call for this could look like this, where `euler-d` is a function you'll have to write:

   ```
   (cont-frac (lambda (x) 1.0)
              euler-d
              100)
   ```

   Then add 2 and compare it to e.

3. J.H.Lambert showed in 1770 that the tangent function could be computed with the continued fraction

   $$\tan(x) = \cfrac{x}{1 - \cfrac{x^2}{3 - \cfrac{x^2}{5 - \cfrac{x^2}{7 - \ddots}}}}$$

   Observe the minus signs, and that the first x is not squared.

   Use your continued fraction procedure to write your own approximate tangent procedure, `mytan`, and compare your approximations for several angles with Scheme's builtin `tan` function.

   The main function call for this should look like:

   ```
   (cont-frac (make-lambert-n x)
              lambert-d
              100)
   ```

   where `lambert-d` and `make-lambert-n` are functions you'll have to write. `make-lambert-n` is, of course, a function of a number, x, that returns a function of the index, i. The returned function, in turn, will return x if $i = 1$ and $-x^2$ otherwise.

   Compare the result to $\tan(x)$, for several values of x.