CS61A – Homework 6.1	**Kurt Meinz**
University of California, Berkeley	Summer 2002

Topic: Streams

Lectures: Monday 7/29, Tuesday 7/30

Reading: Abelson & Sussman, Section 3.5.1–3, 3.5.5

Homework due 10 AM Monday, 8/5:

1. Abelson & Sussman, exercises 3.50, 3.51, 3.52, 3.53, 3.54, 3.55, 3.56, 3.64, 3.66, 3.68

2. Write and test two functions to manipulate nonnegative proper fractions. The first function, `fract-stream`, will take as its argument a list of two nonnegative integers, the numerator and the denominator, in which the numerator is less than the denominator. It will return an infinite stream of decimal digits representing the decimal expansion of the fraction. The second function, `approximation`, will take two arguments: a fraction stream and a nonnegative integer `numdigits`. It will return a list (not a stream) containing the first `numdigits` digits of the decimal expansion.
`(fract-stream '(1 7))` should return the stream representing the decimal expansion of $\frac{1}{7}$, which is 0.142857142857142857...
`(stream-car (fract-stream '(1 7)))` should return 1.
`(stream-car (stream-cdr (stream-cdr (fract-stream '(1 7)))))` should return 2.
`(approximation (fract-stream '(1 7)) 4)` should return `(1 4 2 8)`.
`(approximation (fract-stream '(1 2)) 4)` should return `(5 0 0 0)`.

Extra for experts:

1. Do exercises 3.59–3.62.

2. Consider this procedure:

```
(define (hanoi-stream n)
  (if (= n 0)
      the-empty-stream
      (stream-append (hanoi-stream (- n 1))
                     (cons-stream n (hanoi-stream (- n 1))))))
```

It generates finite streams; here are the first few values:

```
(hanoi-stream 1)    (1)
(hanoi-stream 2)    (1 2 1)
(hanoi-stream 3)    (1 2 1 3 1 2 1)
(hanoi-stream 4)    (1 2 1 3 1 2 1 4 1 2 1 3 1 2 1)
```

Notice that each of these starts with the same values as the one above it, followed by some more values. There is no reason why this pattern can't be continued to generate an infinite stream whose first $2^n - 1$ elements are `(hanoi-stream n)`. Generate this stream.

Unix feature of the assignment: `diff`, `wc`

Emacs feature of the assignment: `M-a`, `M-e`, `M-{`, `M-}`, `M-<`, `M->` (move around buffer)