Streams
1. Define a stream ones with an infinite number of 1s
   (define ones (cons-stream 1 ones))

2. Define a procedure (integers-starting n) that takes in a number n and, when run, returns a cons pair whose car is n, and
   whose cdr is a procedure that, when run with no arguments, does the same thing for n+1.
   (define (integers-starting n)
     (cons-stream n (integers-starting (+ n 1))))

3. What do each of these define?
   (define s1 (add-stream (stream-map (lambda(x) (* x 2)) s1) s1))
   unbound variable
   (define s2
     (cons-stream 1
       (add-stream (stream-map (lambda(x) (* x 2)) s2) s2)))
   powers of 3
   (define s3
     (cons-stream 1
       (stream-filter (lambda(x) (not (= x 1))) s3)))
   infinite loop
   (define s4
     (cons-stream 1
       (cons-stream 2
         (stream-filter (lambda(x) (not (= x 1))) s4))))
   (1 2 2 2 2 ...)
   (define s5
     (cons-stream 1
       (add-streams s5 integers)))
   (1 2 4 7 11 16 ... )

4. Define facts without defining any procedures; the stream should be a stream of 1!, 2!, 3!, 4!, etc. More specifically, it returns a
   stream with elements (1 2 6 24 )
   (define facts
     (cons-stream 1
       (stream-map * (stream-cdr integers) facts)))

5. Define a procedure, (list-to-stream ls) that takes in a list and converts it into a stream.
   (define (list->stream ls)
     (cond ((null? ls) the-empty-stream)
       (else (cons-stream (car ls) (list->stream (cdr ls))))))

6. Define a procedure (lists-starting n) that takes in n and returns a stream containing (n), (n n+1), (n n+1 n+2), etc. For example,
   (lists-starting 1) returns a stream containing (1) (1 2) (1 2 3) (1 2 3 4)
   (define (lists-starting n)
     (cons-stream (list n)
       (stream-map (lambda(ls) (cons n ls)) (lists-starting (+ n 1)))))
9. Define a procedure (make-alternating s) that takes in a stream of positive numbers and alternate their signs. So (make-alternating ones) returns (1 -1 1 -1) and (make-alternating integers) returns (1 -2 3 -4). Assume s is an infinite stream.

    (define (make-alternating s)
        (cons-stream (stream-car s)
                    (cons-stream (* -1 (stream-car (stream-cdr s)))
                        (make-alternating (stream-cdr (stream-cdr s))))))

10. Construct a stream all-integers that includes 0 and both the negative and positive integers. Hint: You should use make-alternating

    (define all-integers
        (interleave (make-alternating (integers-starting 0))
                    (make-alternating (integers-starting 1))))

11. Given streams ones, twos, threes and fours, write down the first ten elements of: (interleave ones (interleave twos (interleave threes fours)))

    (interleave threes fours) ==> (3 4 3 4 3 4 ...)
    (interleave twos threes-fours) ==> (2 3 2 4 2 3 2 4 ...)
    (interleave ones twos-threes-fours) ==> (1 2 1 3 1 2 1 4 1 2 1 3 ...