Q1 (Streams): Define a function `zero-through-three` that returns a stream which cycles through the values 0, 1, 2, 3 every time it's called, starting with 0 (Racket has a function `remainder` that may be useful; e.g. `(remainder x n)` returns x modulo n).

Q2 (Streams): Define a function `zero-through-n` that takes a number n and returns a stream which cycles through the values 0, 1, 2, ..., n-1 every time it's called, starting with 0. You may assume n is non-negative.

Q3 (2019 Summer Final Q2 (a)):

2. (Thunks and Streams – 18 points) As in class, we define a stream to be a thunk that when called returns a pair where the cdr of the pair is a stream. We assume all streams are pure (no printing, mutation, etc.). Assume the following streams are defined:

   - `nats = 1, 2, 3, 4, 5, ...` (the natural numbers)
   - `evens = 2, 4, 6, 8, 10, ...` (the positive even integers)
   - `negs = -1, -2, -3, -4, -5, ...` (the negative integers)

a) Write a Racket function `weave-streams` that takes two stream arguments, s1 and s2, and returns a stream. The resulting stream should contain alternating elements from the two argument streams. That is, the odd-numbered elements of the result stream should be elements (in order) from s1, and the even-numbered elements of the result stream should be elements (in order) from s2.

For example, `(weave-streams nats negs)` would represent 1, -1, 2, -2, 3, -3, ....