CSCI 301, Lab # 4
'Winter, 2017

Due: Your program, named `lab04.rkt`, must be submitted to Canvas before midnight, Tuesday, February 7.

Composition: Let f and g be two one-argument functions. The *composition* f after g, written $f \circ g$, is defined as $(f \circ g)(x) = f(g(x))$.

Programming composition: Define a procedure `compose` that implements composition. For example:

```
((compose square add1) 5) => 36
((compose add1 square) 5) => 26
```

Repeated application: If f is a numerical function and n is a non-negative integer, then we can form the nth repeated application of f, which is defined to be the function whose value at x is $f(f(\ldots(f(x))\ldots))$, where f is repeated n times. There seems to be no standard notation for this, but $f^{\circ n}(x)$ is popular.

For example, if f is the function $f(x) = x + 1$ then the nth repeated application of f is the function $f^{\circ n}(x) = x + n$. If f is the operation of squaring a number, then the nth repeated application of f is the function that raises its argument to the 2^nth power.

Programming repeated application: Using `compose` from the previous problem, write a procedure that takes as inputs a procedure that computes f and a positive integer n and returns the procedure that computes the nth repeated application of f. Your procedure should be able to be used as follows:

```
((repeated square 2) 5)  => 625
((repeated square 3) 5) => 390625
((repeated add1 50) 50) => 100
```

Recursive and iterative: Write both a recursive and an iterative version of the `repeated` procedure. The iterative version should use a tail-recursive function.