```
(define zero (lambda (f) (lambda (x) x)))
(define (add-1 n)
  (lambda (f) (lambda (x) (f ((n f) x)))))
```

Evaluate `(add-1 zero)` using the substitution method.

```
(add-1 zero)
(lambda (f) (lambda (x) (f (((lambda (f) (lambda (x) x)) f) x))))
(lambda (f) (lambda (x) (f ((lambda (x) x) x))))
(lambda (f) (lambda (x) (f x)))
```

```
(define one (lambda (f) (lambda (x) (f x))))
```

Now evaluate `(add-1 one)` to get the definition of `two`.

```
(add-1 one)
(lambda (f) (lambda (x)
  (f (((lambda (f) (lambda (x) (f x))) f) x))))
(lambda (f) (lambda (x)
  (f ((lambda (x) (f x)) x))))
(lambda (f) (lambda (x) (f (f x))))
```

```
(define two (lambda (f) (lambda (x) (f (f x)))))
```

```
(define (+ a b)
  (lambda (f) (lambda (x) ((a f) ((b f) x)))))
```

The addition procedure works by substituting the body of the second term into the `x` parameter of the first term.