```
(define (+ a b)
  (if (= a 0) b (inc (+ (dec a) b))))

(+ 4 5)
(if (= 4 0) 5 (inc (+ (dec 4) 5)))
(inc (+ 3 5))
(inc (if (= 3 0) 5 (inc (+ (dec 3) 5))))
(inc (inc (+ 2 5)))
(inc (inc (if (= 2 0) 5 (inc (+ (dec 2) 5)))))
(inc (inc (inc (+ 1 5))))
(inc (inc (inc (if (= 1 0) 5 (inc (+ (dec 1) 5))))))
(inc (inc (inc (inc (+ 0 5)))))
(inc (inc (inc (inc (if (= 0 0) 5 (inc (+ (dec 0) 5)))))))
(inc (inc (inc (inc 5))))
(inc (inc (inc 6)))
(inc (inc 7))
(inc 8)
```

9

The above process is recursive because it builds up a chain of deferred operations (`inc` procedure calls) before contracting when those operations are performed, incrementing until only the answer remains.

```
(define (+ a b)
  (if (= a 0) b (+ (dec a) (inc b))))

(+ 4 5)
(if (= 4 0) 5 (+ (dec 4) (inc 5)))
(+ 3 6)
(if (= 3 0) 6 (+ (dec 3) (inc 6)))
(+ 2 7)
(if (= 2 0) 7 (+ (dec 2) (inc 7)))
(+ 1 8)
(if (= 1 0) 8 (+ (dec 1) (inc 8)))
(+ 0 9)
(if (= 0 0) 9 (+ (dec 0) (inc 9)))
```

9

The above process is iterative because its state can be summarized by two state variables (`a` and `b`) and a fixed rule to update them (apply `dec` to `a` and `inc` to `b`). There is also an end test to return `b` when `a` is zero.