SICP Exercise 1-6

Samuel Lair

January 19, 2020

1 Solution

This exercise is equivalent to asking what will happen when Scheme interprets the following program:

```
(define (average x y)
    (/ (+ x y) 2))
(define (improve guess x)
    (average guess (/ x guess))
(define (square x) (* x x))
(define (abs x)
    (cond ((< x 0) (- x))
          (else x)))
(define (good-enough? guess x)
    (< (abs (- (square guess) x)) 0.001)
(define (new-if predicate then-clause else-clause)
    (cond (predicate then-clause)
        (else else-clause)))
(define (sqrt-iter guess x)
    (new-if (good-enough? guess x)
        guess
        (sqrt-iter (improve guess x)
                   x)))
(sqrt-iter 1 2)
```

Scheme uses applicative-order evaluation. The built-in if is a special form where only the branch chosen by the value of the predicate is evaluated. However, new-if is an operator. Therefore, both of the operands will evaluated regardless of the value of the predicate.

This poses a problem because sqrt-iter is a recursive operator. Even when the base case is satisfied (i.e. the guess is good-enough), the Scheme interpeter will still evaluate yet another sqrt-iter combination. Therefore, the Scheme interpreter is stuck in an infinite loop when it evaluates (sqrt-iter 1 2).

However, this problem is solved when if is used instead of new-if. In this version of the program, when the guess is good-enough, guess is evaluated without evaluating the recursive combination.