My SICP exercises

Evgeny Markin

2023

Contents

Preface

Exercises are from "Structure and Interpretation of Computer Programs" by Abelson and Sussmans. Most of those exercises are just the programs that need to be written, but some require you to write something down as a text.

Chapter 1

Building Abstractions with Procedures

1.1

Below is a sequence of expressions. What is the result printed by the interpreter in response to each ex- pression? Assume that the sequence is to be evaluated in the order in which it is presented.

```
10 -> 10
(+ 5 3 4) -> 12
(- 9 1) -> 8
(/ 6 2) -> 3
(+ (* 2 4) (- 4 6)) -> 6
(define a 3) -> 3
(define b (+ a 1)) -> 4
(+ a b (* a b)) -> 19
(= a b) -> #f
(if (and (> b a) (< b (* a b)))
b
a) -> 4
(cond ((= a 4) 6)
      ((= b 4) (+ 6 7 a))
      (else 25)
) -> 16
(+ 2 (if (> b a) b a)) -> 6
(* (cond ((> a b) a)
         ((< a b) b)
         (else -1))
```

```
(+ a 1)) -> 16
```

Verified most of them in guile.

1.2

Translate the following expression into prefix form

$$\frac{5+4+(2-(3-(6+\frac{4}{5})))}{3(6-2)(2-7)}$$

```
(/ (+ 5 4 (- 2 (- 3 (+ 6 (/ 4 5))))) (* 3 (- 6 2) (- 2 7))
```

1.4

Observe that our model for combinations whose operators are compound expres- sions. Use this observation to describe the behavior of the following procedure:

```
(define (a-plus-abs-b a b)
  ((if (> b 0) + -) a b)
)
```

Inside if returns plus or minus depending on the value of b, thus this funtion returns

$$a+b$$

in case if $b > 0$ and

$$a-b$$

otherwise, making it effectively equivalent to

$$a+|b|$$

1.5

Ben Bitdiddle has invented a test to determine whether the interpreter he is faced with is using applicative- order evaluation or normal-order evaluation. He defines the following two procedures:

```
(define (p) (p))
(define (test x y)
(if (= x 0) 0 y))
```

Then he evaluates the expression

```
(test 0 (p))
```

What behavior will Ben observe with an interpreter that uses applicative-order evaluation? What behavior will he observe with an interpreter that uses normal-order evaluation? Explain your answer. (Assume that the evaluation rule for the special form if is the same whether the interpreter is using normal or applicative order: Thee predicate expression is evaluated first, and the result determines whether to evaluate the consequent or the alternative expression.)

If the interpreter uses the applicative-order evaluation, then it would firstly evaluate the expressions in the test, before applying test. In this case we go into an infinite loop, since (p) produces anoter (p).

If the interpretes uses the normal-order evaluateion, then it would apply the test procedure, then evaluated if and returned with an answer.