Excercise 1.1

Josh H

January 2, 2024

The expressions to be evaluated are `verbatim`. The response printed by the interpreter is *italicized*. When appropriate, we will include the value of the expression at the bottom before the interpreter prints its response.

(1) The expression `10` is a primitive expression, and is the value of the number the numeral names.

```
10
```

10

(2) The combination is evaluated by having its subexpressions evaluated first. The operator is evaluated, which is the primitive procedure `+`, and the operands are evaluated in the same way as (1). The application of `+` to the arguments is handled by the interpreter, and we are done.

```
(+ 5 3 4)
12
```

12

(3) This is similar to (2).

```
(- 9 1)
8
```

8

(4) Similar to (2).

```
(/ 6 2)
3
```

3

(5) From Section 1.1.5, we appear to evaluate subexpressions simultaneously. We show this, where each subexpression is evaluated at once, one step at a time, and on a new line.

```
(+ (* 2 4) (- 4 6))
(+ 8 -2)
6
```

6

(6) The authors do not show the interpreter's response to `define`. With mit-scheme 12.1, the interpreter prints the name.

```
(define a 3)
```

a

(7) Similar to (6).

```
(define b (+ a 1))
```

b

(8) Similar to (5).

```
(+ a b (* a b))
(+ 3 (+ a 1) (* 3 (+ a 1)))
(+ 3 (+ 3 1) (* 3 (+ 3 1)))
(+ 3 4 (* 3 4))
(+ 3 4 12)
19
```

19

(9) The primitive procedure `=` is a predicate, so the value of the expression it is an argument of when evaluated is either true or false. True and false are written as `#t` and `#f`, respectively.

```
(= a b)
(= 3 (+ a 1))
(= 3 (+ 3 1))
(= 3 4)
#f
```

#f

(10) The authors have not shown the evaluation of a predicate. We assume that the symbols `#t` and `#f` remain in their logical position to show the full evaluation of the expression.

```
(if (and (> b a) (< b (* a b)))
    b
    a)

(if (and (> (+ a 1) 3) (< b (* a b)))
    b
    a)

(if (and (> (+ 3 1) 3) (< b (* a b)))
    b
    a)

(if (and (> 4 3) (< b (* a b)))
    b
    a)

(if (and #t (< b (* a b)))
    b
    a)

(if (and #t (< (+ a 1) (* 3 (+ a 1))))
    b
    a)

(if (and #t (< (+ 3 1) (* 3 (+ 3 1))))
    b
    a)

(if (and #t (< 4 (* 3 4)))
    b
    a)

(if (and #t (< 4 12))
    b
    a)

(if (and #t #t)
    b
    a)
```

```
(if #t
    b
    a)

(if #t
    (+ a 1)
    a)

(if #t
    (+ 3 1)
    a)

(if #t
    4
    a)

4
```

4

(11) Similar to (10).

```
(cond ((= a 4) 6)
      ((= b 4) (+ 6 7 a))
      (else 25))

(cond ((= 3 4) 6)
      ((= b 4) (+ 6 7 a))
      (else 25))

(cond (#f 6)
      ((= b 4) (+ 6 7 a))
      (else 25))

(cond (#f 6)
      ((= (+ a 1) 4) (+ 6 7 a))
      (else 25))

(cond (#f 6)
      ((= (+ 3 1) 4) (+ 6 7 a))
```

```
      (else 25))

(cond (#f 6)
      ((= 4 4) (+ 6 7 a))
      (else 25))

(cond (#f 6)
      (#t (+ 6 7 a))
      (else 25))

(cond (#f 6)
      (#t (+ 6 7 3))
      (else 25))

(cond (#f 6)
      (#t 16)
      (else 25))

16
```

16

(12) Similar to (11).

```
(+ 2 (if (> b a) b a))
(+ 2 (if (> (+ a 1) 3) b a))
(+ 2 (if (> (+ 3 1) 3) b a))
(+ 2 (if (> 4 3) b a))
(+ 2 (if #t b a))
(+ 2 (if #t (+ a 1) a))
(+ 2 (if #t (+ 3 1) a))
(+ 2 (if #t 4 a))
(+ 2 4)
6
```

6

(13) Similar to (11).

```
(* (cond ((> a b) a)
 ((< a b) b)
 (else -1))
   (+ a 1))
```

```
(* (cond ((> 3 (+ a 1)) a)
 ((< a b) b)
 (else -1))
   (+ 3 1))

(* (cond ((> 3 (+ 3 1)) a)
 ((< a b) b)
 (else -1))
   4)

(* (cond ((> 3 4) a)
 ((< a b) b)
 (else -1))
   4)

(* (cond (#f a)
 ((< a b) b)
 (else -1))
   4)

(* (cond (#f a)
 ((< 3 (+ a 1)) b)
 (else -1))
   4)

(* (cond (#f a)
 ((< 3 (+ 3 1)) b)
 (else -1))
   4)

(* (cond (#f a)
 ((< 3 4) b)
 (else -1))
   4)

(* (cond (#f a)
 (#t b)
 (else -1))
```

```
   4)

(* (cond (#f a)
 (#t (+ a 1))
 (else -1))
   4)

(* (cond (#f a)
 (#t (+ 3 1))
 (else -1))
   4)

(* (cond (#f a)
 (#t 4)
 (else -1))
   4)

(* 4
   4)
```

16

16