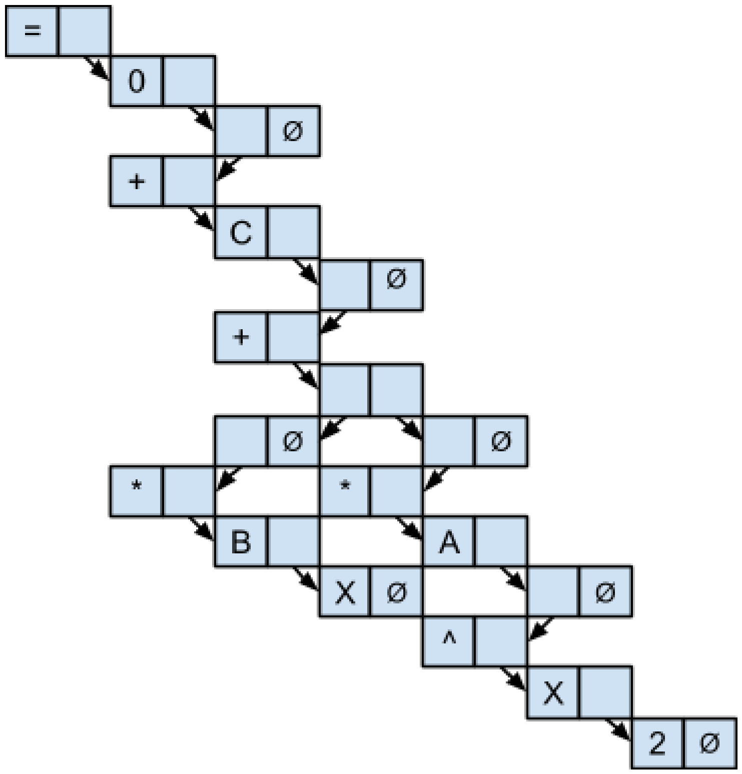
```
1.1.1: (ATOM . NIL)
1.1.2: ((LISP . NIL) . NIL)
1.1.3: (((MORE . (YET . NIL)) . NIL) . NIL)
1.1.4: (HOW . (ABOUT . (THIS . NIL)))
1.1.5: (DONT . ((GET . ((FOOLED . NIL) . NIL)) . NIL))

1.2.1: (X1)
1.2.2: (NIL X1)
1.2.3: (KNOW THY SELF)
1.2.4: ((BEFORE AND AFTER))
1.2.5: (A ((B C)))

1.3.1: ((W . NIL) . NIL)
1.3.2: (NIL . (NIL . (NIL . NIL)))
1.3.3: (((NEST . NIL) . NIL) . NIL)
1.3.4: (W . ((B . NIL) . (B . NIL)))
1.3.5: (((A . NIL) . (B . NIL)).((C . NIL) . (D . NIL)))

1.4.1: (= 0 (+ C (+ (* B X) (* A (^ X 2)))))
1.4.2:
(= . (0 . ((+ . (C . ((+ . (((* . (B . (X . NIL))) . NIL) . ((* . (A . ((^ . (X . (2 . NIL))) . NIL)))
. NIL))) . NIL))) . NIL)))
1.4.3:
```



```
2.1.1: LEFT
2.1.2: RIGHT
2.1.3: (LEFT . RIGHT)
2.1.4: A
2.1.5: (A)
2.1.6: A
2.1.7: (A . B)
```

```
2.1.8: (SENTENCE IS A LIST)
2.1.9: ((ABOUT THIS))
2.1.10: ((DOT . PAIR2))

2.2.1: CADAR
2.2.2: CAAR
2.2.3: CADR
2.2.4: CDR
2.2.5: CADDR
2.2.6: CADDDR
2.2.7: CAAR
2.2.8: CDAR
2.2.9: CAAADR
2.2.10: CAADR
2.2.11: CADADR

3.1: ATOM
3.2: (LIST)
3.3: THREE
3.4: (ELEMENT LIST)
3.5: (VERY GOOD)
3.6: (ONE (THEN . ANOTHER))
3.7: B
3.8: NIL
3.9: 3
3.10: 3
3.11: ERROR: TOO MANY ARGUMENTS
3.12: ERROR: TOO MANY ARGUMENTS
3.13: ALPHA
3.14: BETA
3.15: FIRST

4.1: (F ‘1) -> 1
4.2:
(F ‘(1 2 3)) -> (CONS (F (2 3)) (F 1)) -> (CONS (CONS (F (3)) (F 2)) 1) -> (CONS (CONS (CONS (F NIL) (F
3)) 2) 1) -> (CONS (CONS (CONS NIL 3) 2) 1) -> (CONS (CONS (NIL . 3) 2) 1) -> CONS ((NIL . 3) . 2) 1)
-> (((NIL . 3) . 2) . 1)
4.3:
(F ‘(1 . (2 . 3))) -> (CONS (F (2 . 3)) (F 1)) -> (CONS (CONS (F 3) (F 2)) 1) -> (CONS (CONS 3 2) 1) ->
(CONS (3 . 2) 1) -> ((3 . 2) . 1)
4.4: An appropriate name for this function is REVERSE.

5.1.1:
(DEFUN APPEND2 (LIST ATOM)
  (COND
    ((ATOM LIST) (COND
      ((EQ NIL LIST) ATOM)
      (T (ERROR "ERROR: A proper list must end with NIL."))))
    (T (CONS (CAR LIST) (APPEND2 (CDR LIST) ATOM)))))
5.1.2:
(APPEND2 NIL ‘9) -> 9
(APPEND2 ‘1 ‘9) -> ERROR: A proper list must end with NIL.
(APPEND2 ‘(1) ‘9) -> (1 . 9)

5.2.1:
(DEFUN MEMBER2 (ATOM LIST)
  (COND
    ((ATOM LIST) (EQ LIST ATOM))
    (T (OR (MEMBER2 ATOM (CAR LIST)) (MEMBER2 ATOM (CDR LIST))))))
5.2.2:
(MEMBER2 NIL ‘(1 2 3)) -> T
(MEMBER2 3 ‘(1 2 3)) -> T
(MEMBER2 4 ‘(1 2 3)) -> F

5.3.1:
(DEFUN INTERSECT2 (LIST1 LIST2)
  (COND
    ((ATOM LIST1) NIL)
    ((MEMBER2 (CAR LIST1) LIST2) (CONS (CAR LIST1) (INTERSECT2 (CDR LIST1) LIST2)))
    (T (INTERSECT2 (CDR LIST1) LIST2))))
5.3.2:
(INTERSECT2 NIL ‘(1 2 3)) -> NIL
(INTERSECT2 ‘(1 2 3) ‘(2 3 4)) -> (2 3)
(INTERSECT2 ‘(1 2) ‘(3 4)) -> NIL

5.4.1:
(DEFUN UNION2 (LIST1 LIST2)
  (COND
    ((ATOM LIST1) LIST2)
    ((NOT (MEMBER2 (CAR LIST1) LIST2)) (UNION2 (CDR LIST1) (APPEND2 LIST2 (LIST (CAR LIST1)))))
    (T (UNION2 (CDR LIST1) LIST2))))
5.4.2:
(UNION '(1 2) '(3 4)) -> (1 2 3 4)
(UNION '(1 2) '(2 3 4)) -> (1 2 3 4)

5.5.1:
(DEFUN SORT2 (LIST)
  (COND
    ((ATOM LIST) NIL)
    (T (LET ((X (APPLY 'MIN LIST)))
      (CONS X (SORT2 (REMOVE x LIST)))))))
5.5.2:
(SORT2 '(3 7 2 9)) -> (2 3 7 9)
```