Scheme

New procedures

1. `(cons a b)` - Makes a cons-cell (pair) from a and b
2. `(car c)` - extracts the value of the first part of the pair
3. `(cdr c)` - extracts the value of the second part of the pair
4. `(c da ada c r c)` - shortcuts
5. `(list a b c ...)` - builds a list of the arguments to the procedure
6. `(adjoin a lst)?` - doesn’t exist (use `cons`)
7. `(list-ref lst n)` - returns the nth element of lst
8. `(append l1 l2)` - makes a new list containing the elements of both lists

Problems

1. Draw box-and-pointer for the values of the following expressions. Also give the printed representation.

   `(cons 1 2)`

   `(cons 1 (cons 3 (cons 5 nil)))`

   `(cons (cons (cons 3 2) (cons 1 0)) nil)`

   `(cons 0 (list 1 2))`

   `(list (cons 1 2) (list 4 5) 3)`
2. Write expressions whose values will print out like the following.

(1 2 3)

(1 2 . 3)

(((1 2) (3 4) (5 6)))

3. Write expressions using car and cdr that will return 4 when the lst is bound to the following values:

(7 6 5 4 3 2 1)

(((7) (6 5 4) (3 2) 1)

(7 (6 (5 (4 (3 (2 (1))))))

(7 (((6 5 ((4)) 3) 2) 1)

Abstraction

Suppose you’re working for the registrar, and she asks you to develop a scheme system to keep track of each student’s registration...

Structures?
Procedures?