1)

```
7 + (2 * -1/3) + -10.7       => (+ (+ (* 2 -1/3) 7) -10.7)
(7/3 * 5/9) ÷ (5/8 - 2/3)    => (/ (* 7/3 5/9) (- 5/8 2/3))
1 + 3 ÷ (2 + 1 ÷ (5 + 1/2))  => (+ 1 (/ 3 (+ 2 (/ 1 (+ 5 1/2)))))
1 × -2 × 3 × -4 × 5 × -6 × 7 => (* 1 (* -2 (* 3 (* -4 (* 5 (* -6
7))))))
```

2)

```
(cons 'car '+) = (car . +) ; crea una lista impropia
(list 'esto '(es muy fácil)) = (esto (es muy fácil)) ; crea una
lista propia
(cons 'pero '(se está complicando...)) = (pero se está
complicando...) ; si el segundo argumento es una lista propia, la
lista resultante sera propia
(cons '(y ahora no se que ) 'hizo) = ((y ahora no se que ) . hizo) ;
si el segundo argumento es un elemento, la lista resultante sera
impropia
(quote (+ 7 2)) = (+ 7 2) ; ignora los procedimientos dentro del
argumento
(cons '+ '(10 3)) = (+ 10 3) ; como el segundo argumento es una
lista, se crea una lista propia
(car '(+ 10 3)) = + ; devuelve la cabecera de la lista
(cdr '(+ 10 3)) = (10 3) ; devuelve la lista, sin la cabecera
cons = #<procedure:mcons> ; informa que es un procedimiento
(quote (cons (car (cdr (7 4))))) = (cons (car (cdr (7 4)))) ; ignora
los procedimientos dentro del argumento
(quote cons) = cons ; ignora los procedimientos dentro del argumento
(car (quote (quote cons))) = quote ; quote arma la lista (quote
cons) y car toma la cabecera de dicha lista
(+ 2 3) = 5 ; realiza la operacion
(+ '2 '3) = 5 ; interpreta las cadenas segun el primer argumento del
procedimiento
(+ (car '(2 3)) (car (cdr '(2 3)))) = 5 ; efectua los procedimientos
de las listas, y ejecuta la operacion
((car (list + - * /)) 2 3) = 5 ; toma la cabecera de la lista y
efectua la operacion
```

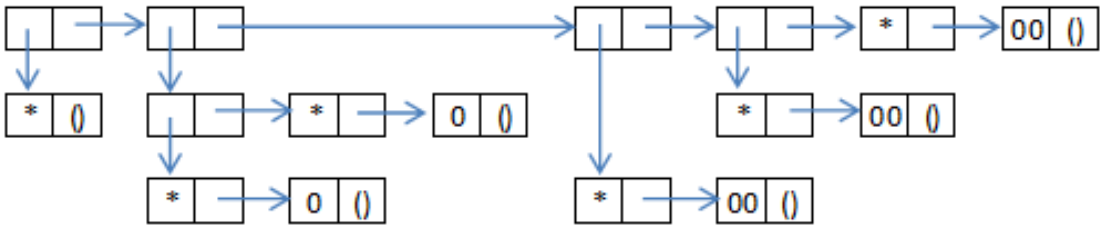
3)

```
(car (cdr (car '((a b) (c d))))) = b
(car (car (cdr '((a b) (c d))))) = c
(car (cdr (car (cdr '((a b) (c d)))))) = d
```

4)

```
(list '(((b . c) . (d . e)) a f))
```

5)



6)

```
(car (car '((a b) (c d)))) = a
(cdr (car '((a b) (c d)))) = (b)
(car (cdr '((a b) (c d)))) = (c d)
(cdr (cdr '((a b) (c d)))) = ()
```

7)

```
((car (cdr (cdr (list + - * /)))) 5 5)
```

(list + - * /) crea la lista de prmitivas = (+ - * /)
(cdr (list + - * /)) toma la lista sin la cabecera = (- * /)
(cdr (cdr (list + - * /))) toma la lista sin la cabecera de la operacion anterior = (* /)
(car (cdr (cdr (list + - * /)))) toma la cabecera de la lista de la operacion anterior = *
((car (cdr (cdr (list + - * /)))) 5 5) efectua la operacion con los argumentos = (* 5 5) = 25