**Cabrera Herrera Nathaniel**

1.a) (first (rest (rest (rest (rest '(((1 2) 3) 4 (5 (6)) a (b c) d (e (f g))))) )))

1.b) (\* 60 60 24 366)

1.c) (defun lol (x y) (and (/= x 0) (<= x y)) )

1.d) (defun lol (a b c) (cons (/(+ (- 0 b) (sqrt(- (expt b 2)(\* 4 a c))))( \* 2 a)) (/(- (- 0 b) (sqrt(- (expt b 2)(\* 4 a c))))( \* 2 a)) ))

2.a) (+ (\* 2 4) (- 6 8))

2.b) (/ (+ 5 -3 4) (+ 6 (/ 2 5)) )

2.c) (sqrt (/ (+ (- (+ -4 -3/8)) 1.4502 ) (- (expt 1 (expt (- 3 5) 1/3))) ) )

2.d) (expt (/ (expt (/ 65.402 (sqrt -1) ) 1/5) .17) 1/7)

3.a) (TWO)

Es el resultado del hacer FIRST y después REST (en su equivalente) lo que retorna una lista con el segundo miembro del primer elemento de la lista que en este caso es otra lista.

3.b) ((EVA LISA) KARL SVEN EVA LISA KARL SVEN)

Lo primero que pasa es que la macro CONS introduce la lista (eva lisa) como primer elemento de la lista (Karl sven) y posteriormente une el contenido de esa lista con el contenido de las otras dos listas en una sola lista.

3.c) (EVA GITAN LISA GITAN KARIN)

Sustituye el átomo BIRGITTA por GITAN en la lista

3.d) (EVA LISA ANNA)

Elimina los átomos contenidos en la lista que coincidan con el átomo provisto.

3.e) (KARL ADAM NILSSON)

Conserva una lista con el número de átomos que se indiquen con el parámetro.

3.f) C

Retorna el n-simo atomo contenido de la lista de derecha a izquierda empezando el conteo desde el átomo 0.

3.g) (C D E)

Genera un CDR a partir de un n-simo atomo.

3.h) (C B)

Retorna una lista con los atomos que comparten ambas listas

3.i) (4)

Desglosando el procedimiento pasa lo siguiente

(((((1 2 3) z) y) (x 4)) 7 8 (a b c (5 (6 7 8)))) - Forma Original

((((1 2 3) z) y) (x 4)) - Al aplicar el primer CAR y obtiene una lista con el primer elemento.

((x 4)) - Al aplicar el primer CDR obtiene el ultimo elemento de la lista, que es una lista

(x 4) - Al aplicar el segundo CAR obtiene el primer elemento, que era la única lista no NIL

(4) - Al aplicar el segundo CDR obtiene el ultimo elemento que es el atomo 4 y lo muestra dentro de una lista

4)

5) (zerop <N>) -Esta es una función existente

(defun NoCero (n) (if (/= (- 0 n) 0) "No es 0" "Es 0" )) - Esta es mia

6) (defun Analiza (x)

(if (atom x) (print 'Es\_atomo) (print 'No\_es\_atomo) )

(if (numberp x) (print 'Es\_número) (print 'No\_es\_número) )

(if (listp x) (print 'Es\_una\_lista) (print 'No\_es\_una\_lista))

(if (consp x) (print 'Es\_una\_celda\_de\_construcción) (print 'No\_es\_una\_celda\_de\_Construcción))

(if (equal x NIL) 'Es\_una\_lista\_vacia 'No\_es\_una\_lista\_vacia)

)