TRABAJO PRÁCTICO N° 2
Interprete TLC LISP en LISP
Máquina virtual de LISP

Fecha de entrega:

Escribir la función "evaluar" en Lisp que evalúa expresiones escritas en TLC-Lisp, que recibe dos párametros:

-una expresión escrita en TLC-Lisp
-el ambiente que está representado por una lista donde hay nombres seguido de su valor asociado

y devuelve el valor de la expresión dada.

La función evaluar debe trabajar con su ambiente propio sin buscar asociaciones afuera de su ambiente, por eso decimos que es una virtual-box.
La función evaluar debe imitar a la función val de Lisp y debe estar desarrollada semánticamente como se dió en las clases teóricas de la materia.

(defun evaluar(exp amb) …)

A continuación se muestras ejemplos de invocación de la función evaluar.

; con numeros
(evaluar '2 nil)
> ------> 2
; con valores booleanos true false
(evaluar nil nil)
> ------> nil
(evaluar 't nil)
> ------> t
;asociaciones en el ambiente
(evaluar 'A '(A 2))
> ------> 2
(evaluar 'B '(A 2 B 10)

```
> ------> 10
;la función quote
(evaluar '(quote A) nil)
> ------> A
(evaluar '(quote 1) nil)
> ------> 1
(evaluar '(quote (car a)) nil )
> ------> (car a)
(evaluar '(quote ((2 3) (4 5))) )
> ------> ((2 3) (4 5))
;funciones booleanas  and y or
(evaluar '(and (or t nil) t) nil )
> ------> t
(evaluar '(and (or t nil) (or nil nil)) nil)
> ------> nil
(evaluar '(or (or t nil) (or nil nil )) nil)
>------> t
;Función car + ambiente
(evaluar '(car (list a 2 3)) '(a 100) )
> ------> 100
;Función cdr + ambiente
(evaluar '(cdr (list a b c)) '(a 100 b 99 c 98) )
> ------> (99 98)
;Funciones anónimas lambda
(evaluar '((lambda (x) (* x 2)) 2) nil )
> ------> 4
(evaluar '((lambda (x y) (+ (* x 2) y)) 2 4) nil)
> ------> 8
(evaluar '(lambda (x) (* x 2)) nil)
> ------> (lambda (x) (* x 2))
(evaluar '(mapcar (lambda (x) (cons x (cdr '(3 4 5)))) '(1 2 3)) nil)
> ------> ((1 4 5) (2 4 5)(3 4 5))
;Forma funcional  mapcar
(evaluar '(mapcar 'numberp (quote (4)))) '(t))
(evaluar '(mapcar 'numberp (quote (4 5 6 nil))))
> ------> (t t t nil)
(evaluar '(mapcar 'car (quote ( (2 3) (4 5 ))) ))
> ------> (2 4)
;Funciones definidas en el ambiente
(evaluar '(fact 5) '(fact (lambda(n)(if(eq n 0) 1 (* n (fact (- n 1))))) ) )
> ------> 120
(evaluar '(mapcar 'fact (quote ( 2 3 4 5 ) ))
          '(fact (lambda(n)(if(eq n 0) 1 (* n (fact (- n 1)))))   )
> ------> (2 6 24 120)
```