PARADIGMAS Y LENGUAJES DE PROGRAMACIÓN
- LENGUAJE LISP -

FUNCIÓN ITERATIVA MAPCAR

Pensar y razonar las soluciones antes de intentar programarlas

NOTA:

- Todos los funciones iterativas deben resolverse utilizando la función MAPCAR
- Recordar que para aplicar funciones numéricas, sus argumentos deben ser numéricos. Se deberán evaluar los mismos de ser necesarios
- Las variables nuevas que se definan deben ser locales y tener mensajes descriptivos en su solicitud de ingreso.

Ejercicio Nº 1.
Evaluar las siguientes funciones y determinar su resultado

Función aplicada	**Resultado**
(mapcar 'atom '(a (b) () (()) “AA” 3))	
(mapcar 'listp '(a (b) () (()) "AA" 3))	
(mapcar '> '(5 8 3) '(4 9 2))	
(mapcar '< '(2 8 3) '(4 9 2) '(5 1 7))	
(mapcar '> '(5 8 3) '(4 9) '(3 8 1))	
(mapcar '+ '(1 2) '(3 4) '(2 2))	
(mapcar '- '(1 8) '(3 4) '(2 2 9))	
(mapcar 'cons '(1 2) '((a b) (3 4) ((7 8))))	
(mapcar 'abs '(0 -8 10 3 -2.5 -1/4))	
(mapcar 'length '((1 1 1) () ((8)) (a b)))	
(mapcar 'list '(a b c d))	
(mapcar 'car '((2 3 4) (a b) ((c))))	

Ejercicio Nº 2.
Determinar el resultado que arrojarían las siguientes funciones lambda

Función LISP	**Resultado**
((lambda (X) (if (> (car X) 0) 'POSITIVO)) '(5 6 7))	
((lambda (X) (if (> (car X) 0) 'POSITIVO)) '(-5 6 7))	
((lambda (A) (reverse (cdr A))) '((2 3 4) a b c))	
((lambda (X) (if (numberp (car X)) (* 2 (car X)))) '(5 6 7))	
((lambda (X) (if (and (numberp (car X)) (evenp (car X))) (+ 10 (car X)))) '(4 6 7))	
((lambda (X) (if (and (numberp (car X)) (evenp (car X))) (+ 10 (car X)) 0)) '(7 6 7))	

PARADIGMAS Y LENGUAJES DE PROGRAMACIÓN
- LENGUAJE LISP -

Ejercicio Nº 3.
Definir una función que a partir de una lista ingresada como parámetro devuelva una nueva lista cuyos elementos sean el resultado de evaluar si cada elemento de la lista original es o no un elemento numérico

Ejercicio Nº 4:
Realizar los cambios necesarios en la función definida en el Ejercicio N° 3, de tal manera que el resultado de cada evaluación realizada devuelva la leyenda SI o NO.

Ejercicio Nº 5:
Definir una función que solicite al operador el ingreso de una Lista no vacía y un número entero X, de tal manera que devuelva una lista que asocie los elementos numéricos de la lista original con el resultado de calcular su potencia X.

Ejercicio Nº 6
Definir una función que a partir de una lista heterogénea ingresada por el operador, devuelva una nueva lista formada por sublistas, donde cada sublsita estará formada de la siguiente manera: **(X signo 0)**, donde X será el elemento de la lista original, signo se corresponderá con <, > o = según ese elemento sea mayor, menor o igual a cero.

Ejercicio Nº 7
Definir una función que a partir de una lista heterogénea ingresada como parámetro, devuelva una nueva lista formada por las longitudes de aquellos elementos que sean sublistas.

Ejercicio Nº 8: Definir una función que a partir de dos listas ingresadas como parámetro, devuelva una nueva lista que asocie cada elemento no-numérico de la LISTA1 con el último elemento de la LISTA2.

Ejercicio Nº 9: Definir la función **sumo-ambos** que a partir de 2 Listas no vacías: LISTA1 y LISTA2, devuelva una nueva Lista con el resultado de sumar elemento a elemento, los elementos de la misma posición. (1° elemento de LISTA1 con el 1° elemento de LISTA", 2° elemento de LISTA1 con el 2° elemento de LISTA", ….)

Ejercicio Nº 10: Definir la función **ambos-enteros** que solicite al operador el ingreso de dos Listas no vacías: LISTA1 y LISTA2. La función deberá devolver una nueva Lista con el resultado de evaluar elemento a elemento ambas Listas, indicando en forma de sublista ambos elementos, si coinciden en que sean números enteros.