Nota: Los problemas de Lisp se considerarán solo si funcionan en forma correcta. Para evitar algún inconveniente o error de interpretación, pueden incluso probar sus respuestas. Suba un archivo en word o pdf con las 4 respuestas.

(1.35)

1. La función que aparece abajo **aplana e invierte** una lista a profundidad. Sin embargo, existen algunos errores de paréntesis. Agregue o quite los paréntesis que falten o sobren para que la función corra perfectamente. Abajo se observa un ejemplo. Observe que **NO** debe realizar una nueva función, lo único que debe hacer es solucionar el problema de paréntesis. Sólo si corre perfectamente se considerará válido el ejercicio.

```
> (setq lst '(1 2 3 (4 5 6) 7) lst1 nil)
  (1 2 3 (4 5 6) 7)

> (aplanainv lst)     >lst1
  T                        (7 6 5 4 3 2 1)
```

```
(defun aplanainv (lst)
(cond
((null lst)
(atom (car lst) (push (car lst) lst1) (aplanainv (cdr lst))
(t (aplanainv (car lst) (aplanainv (cdr lst)))
```

Respuesta:

```
(defun aplanainv (lst)
(cond
        ((null lst))
        ((atom (car lst)) (push (car lst) lst1) (aplanainv (cdr lst)))
        (t (aplanainv (car lst)) (aplanainv (cdr lst)))))
```

(1.35)

2. Escriba **una sola instrucción** que permita rotar una lista **lst** una determinada distancia **d**. Observe los siguientes ejemplos.

>(setq lst ‘(a b c d e f) d 1) >(setq lst ‘(a b c d e f) d 3)

(b c d e f a) (d e f a b c)

Respuesta:

Suponiendo todos los valores son aceptados para que no cause ningún error:

(append (nthcdr (mod dist (lenght lst)) lst)) (reverse (nthcdr (- (lenght lst) (mod dist (lenght lst))) (reverse lst)))

La lógica es hacer una nueva lista utilizando el cdr a partir de la dist y el lenght y juntarlos con la original, luego con el reverse quitarlos del inicio y volver a hacer un reverse para que queden en orden.

(1.35)

3. Escriba una sola instrucción que permita obtener una sublista de una lista **lst**. El primer elemento de la lista se encuentra en la posición **n** y el último se encuentra en la posición **m**. n siempre será menor o igual a m. No se debe preocupar por esa razón u otra derivada de n y m. Observe los siguientes ejemplos.

>(setq lst ‘(a b c d e f) n 2 m 4) >(setq lst ‘(a b c d e f) n 3 m 18)

(b c d) (c d e f)

Respuesta:

Para obtener una ‘slice’, asumiendo que m y n están acotados para que no haya ningún error:

(reverse (nthcdr (- (lenght lst) (min ultimo (lenght lst))) (reverse (nthcdr (- (min primero (lenght lst)) 1) lst))))

Siguiendo un poco la lógica del pasado y asumiendo lo de las instrucciones la lógica es utilizando un reverse hacer una lista a partir del cdr del lenght – el min entre el ult y el lenght (por si ult es más grande), luego otro reverse para encontrar

el resultado que quiero entre el min del primero y el lenght y luego restarle uno para iniciar donde quiero y me quede la rebanada bien.

(1.35)

4. Escriba **una sola instrucción** que inserte el elemento **ele** en la posición **n** de la lista **lst**. n nunca tendrá un valor nulo, negativo o será mayor a la longitud de la lista. Observe el siguiente ejemplo y el resultado que aparece abajo.

>(setq lst '(1 2 3 5 6 7) ele 4 n 4)

(1 2 3 4 5 6 7)

>(setq lst '(a b c k l m) ele w pos 2)

(a w b c k l m)

Respuesta:

De acuerdo a las delimitaciones de n en las instrucciones:

(push elemento (nthcdr (- n 1) lst))

Parece muy sencillo, solo es encontrar la posición a partir del cdr de (n -1) y meterlo.