Ejercicios Clase 19-11-2024 – Comisión A

Parte 1. Ejercicios básicos

1) Definir las funciones:
   a) (raiz A) retorna el elemento que está en la raíz del árbol A
   b) (hijo-izq A) retorna el subárbol izquierdo del árbol A
   c) (hijo-der A) retorna el subárbol derecho del árbol A
2) Definir las funciones para recorrer un árbol en preorden, inorden y posorden. Cada función debe retornar una lista con los nodos que se visitan (según el tipo de recorrido).
3) Dados un árbol y un elemento cualquiera, determinar si el elemento pertenece al árbol.
4) Un árbol binario de nivel *n* es *completo, si "cada nodo de nivel n es una hoja y cada nodo de nivel menor que n no tiene subárboles izquierdo y derecho vacíos".* Definir una función de un argumento para determinar si el árbol dado como argumento es completo.

Parte 2. Ejercicios Funciones de Orden Superior (FOS)

1) Definir la función mapeo que recibe como argumento una función de 1 argumento y una lista. Mapeo devuelve una lista que resulta de aplicar la función recibida como argumento a cada elemento de la lista. Ejemplos:

```
(mapeo suma-1 '(1 2 3) )
(2 3 4)
(mapeo car '((1 2 3) ('a 'b 'c) ("hola" "chau")))
(1 'a "hola")
(mapeo cdr '((1 2 3) ('a 'b 'c) ("hola" "chau")))
((2 3) ('b 'c) ("chau"))
```

2) Definir la función filtro. Esta función es una función de orden superior, que tiene como argumentos una: i) función de un argumento que devuelve #t o #f y ii) una lista y retorna una lista cuyos elementos son los elementos de la lista original que devolvieron #t al aplicarle la función recibida como argumento. Ejemplos:

```
> (filtro mayor2 '(1 2 3 4))
4)
> (filtro list? '((1 2 3) 3 (12 3)))
((1 2 3) (12 3))
```

3) Defina la función **(aplica-mapeo LFunciones Lista)** que devuelve una lista que resulta de aplicar cada una de las funciones que son miembros de LFunciones a cada elemento de Lista. Ejemplos:

```
(define suma-1 (lambda (x) (+ x 1)))
(define resta-1 (lambda (x) (- x 1)))
```

```
(define por-2 (lambda (x) (* x 2)))
(aplica-mapeo (list suma-1 por-2) '(1 2 3))
(4 6 8)
(aplica-mapeo (list cdr car) '((1 2 3) ('a 'b 'c) ("hola" "chau")))
(2 'b "chau")
```

4) Defina la función **(aplica-filtro listaFiltros)** que retorne una función cuyo argumento es una lista. En este caso, listaFiltros es una lista de funciones de un argumento que devuelven verdadero o falso. Al evaluar la función resultante con una lista específica se deben aplicar todos los filtros de la lista de funciones para obtener la lista resultante. Ejemplos:

```
(aplica-filtro (list mayor2 menor4))
#<procedure>
((aplica-filtro (list mayor2 menor4)) '(4 2 3 5 1 6))
(3)
```

5) Defina una función de orden superior que recibe como parámetros una función (que retorna verdadero o falso) y una lista. La función **(take-while condicion L)** devuelve una lista con los n primeros elementos de la lista L para los que condicion retorne verdadero. Donde n es la posición del primer elemento para el que la evaluación de condicion retorne falso. Por ejemplo:

```
(define mayor2 (lambda (x) (> x 2)))
(take-while mayor2 '(5 6 7 2 8))
(5 6 7)
(take-while mayor2 '(5 6 7 9 8))
(5 6 7 9 8)
(take-while mayor2 '(1 2 3 3 2))
()
```

6) De manera similar al punto anterior, defina la función **(drop-while condition L)** que devuelve la lista L a la que se le eliminaron los primeros elementos que satisfacen la condición (esto es aquellos elementos que al aplicarle la función condition dan verdadero). Ejemplos:

```
(define menor4 (lambda (x) (< x 4)))
(drop-while menor4 '(2 3 1 5 6 ))
(5 6)
(drop-while menor4 '(2 3 1 5 6 1))
(5 6 1)
(drop-while menor4 '(5 6 1 2 3))
(5 6 1 2 3)
```

7) Dada una función de un argumento f y un entero positivo n, se le solicita que defina la función de orden superior **(repetir f n)**, la cual retorna una función de un argumento que aplica n veces la función f . Ejemplos

```
(define cua (lambda (x) (* x x))
((repetir cua 1) 2)
 4
((repetir cua 3) 2)
 256
```

8) Ejercicio 8

   a) Se le solicita que defina la función *Aplica* de **tres argumentos, una lista, una función f y un número entero n** , que dará como resultado la lista original pero con el elemento que se encuentra en la posición **n** reemplazado por el resultado de aplicar la función **f** al enésimo elemento. Ejemplos:

```
(define suma-1 (lambda (x) (+ x 1)))
(aplica suma-1 '(2 4 9) 1)
(3 4 9)
(aplica suma-1 '(2 4 9) 3)
(2 4 10)
```

   b) Se le solicita que defina una función de dos argumentos, una lista y una función f, aplica-all, que dará como resultado una lista de listas, en la cual el enésimo elemento es la lista original en la cual se ha reemplazado su enésimo elemento por el resultado de aplicar f al mismo. Ejemplos:

```
(define suma-1 (lambda (x) (+ x 1)))
(aplica-all suma-1 '(2 4 9))
((3 4 9) (2 5 9) (2 4 10))
```

Parte 3. Ejercicio de parcial/examen

Resolver los ejercicios 3 y 4 del parcial Nro. 3 correspondiente al cuatrimestre 1 de 2015 (el archive pdf está en la carpeta correspondiente a la clase del 19-11)

Nombre del archivo: *Parcial Funcional 1C-2015.pdf*