

Nombre: _____ **Grupo:** _____

Lenguajes y Paradigmas de Programación

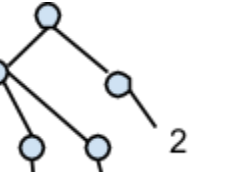

Curso 2013-2014

Segundo parcial - Turno de mañana

Normas importantes

- La puntuación total del examen es de 10 puntos.
- Se debe contestar cada pregunta en las hojas que entregamos. Utiliza las últimas hojas para hacer pruebas. No olvides poner el nombre.
- La duración del examen es de 2 horas.

b) (0,75 puntos) Dadas las siguientes estructuras de datos recursivas, escribe su expresión correspondiente en forma de lista estructurada y las instrucciones, utilizando la barrera de abstracción adecuada, para obtener el elemento 10 de cada una:

Pseudoárbol: 	
Lista estructurada:	
Elemento 10:	
Árbol genérico: 	
Lista estructurada:	
Elemento 10:	

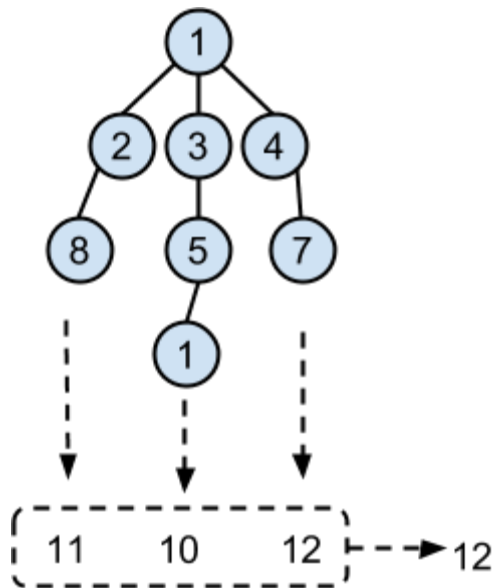
Ejercicio 2 (1,5 puntos)

Implementa en Scheme la función recursiva `suma-listas` que reciba dos listas estructuradas con la misma estructura y devuelva una lista plana con las sumas de todos sus elementos:

```
(suma-listas '(1 2 (3 (4) (5 (6 7)))) 3)
              '(4 2 (1 (8) (7 (2 3)))) 2)) => (5 4 4 12 12 8 10 5)
```

Ejercicio 4 (1,5 puntos)

Utilizando recursión mutua o funciones de orden superior, define en Scheme la función `suma-max-tree` que reciba un árbol genérico y devuelva, de la suma de los nodos de cada rama, aquella que es máxima. Ejemplo:



El tema de clausuras con estado local lo dimos en Scala.

```
val x = 10
val y = 20
def g(f:(Int,Int)=>Int) = {
  val x = 30
  val y = 40
  f(x,y)
}
def bar(x:Int,y:Int) = x+y
val h = bar _
val a = g(h)
```

```

graph TD
    N1((1)) -- h --> N2((2))
    N2 -- f --> N3((3))
    N3 -- a --> N1
    
```

Diagram illustrating a control flow graph (CFG) with three nodes (1, 2, 3) and associated variables and expressions.

- Node 1:** Initial state. Variables: $x: 80$, y . Expression: $\{x+y+z\}$.
- Node 2:** Intermediate state. Variables: $y: 20$. Expression: $x+y+z \rightarrow 110$.
- Node 3:** Intermediate state. Variables: $y: 40$. Expression: $x+y+z \rightarrow 130$.

Control flow edges and labels:

- Edge from Node 1 to Node 2: labeled h .
- Edge from Node 2 to Node 3: labeled f .
- Edge from Node 3 to Node 1: labeled a .

Global environment (Entorno global) variables:

- $z: 10$
- $w1: 110$
- $w2: 130$

Nombre: _____ Grupo: _____

Lenguajes y Paradigmas de Programación

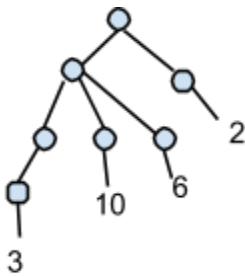
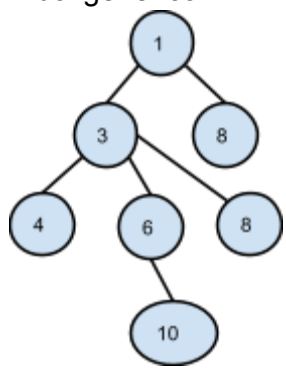
Curso 2013-2014

Segundo parcial - Turno de tarde

Normas importantes

- La puntuación total del examen es de 10 puntos.
- Se debe contestar cada pregunta en las hojas que entregamos. Utiliza las últimas hojas para hacer pruebas. No olvides poner el nombre.
- La duración del examen es de 2 horas.

b) (0,75 puntos) Dadas las siguientes estructuras de datos recursivas, escribe su expresión correspondiente en forma de lista estructurada y las instrucciones, utilizando la barrera de abstracción adecuada, para obtener el elemento 6 de cada una:

<p>Pseudoárbol:</p> 	<p>Lista estructurada:</p> <p>Elemento 6:</p>
<p>Árbol genérico:</p> 	<p>Lista estructurada:</p> <p>Elemento 6:</p>

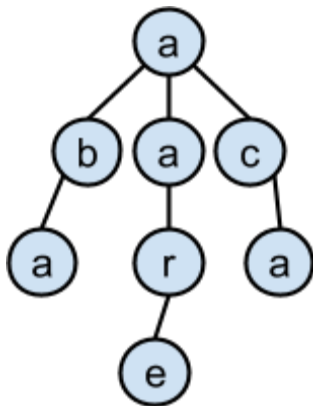
Ejercicio 2 (1,5 puntos)

Implementa en Scheme la función recursiva `compara-listas` que reciba dos listas estructuradas con la misma estructura y devuelva una lista plana con booleanos que indiquen si el elemento de la segunda lista es mayor o igual que el de la primera:

```
(compara-listas '(1 2 (3 (4) (5 (6 7)))) 3)
                '(4 2 (1 (8) (7 (2 3)))) 2)) => (#t #t #f #t #t #f #f #f)
```

Ejercicio 4 (1,5 puntos)

Utilizando recursión mutua o funciones de orden superior, define en Scheme la función `cuenta-elem-tree` que reciba un árbol genérico y un elemento como argumentos, y devuelva el número de veces que aparece el elemento en los nodos del árbol. Ejemplo:



`(cuenta-elem-tree tree 'a) → 4`

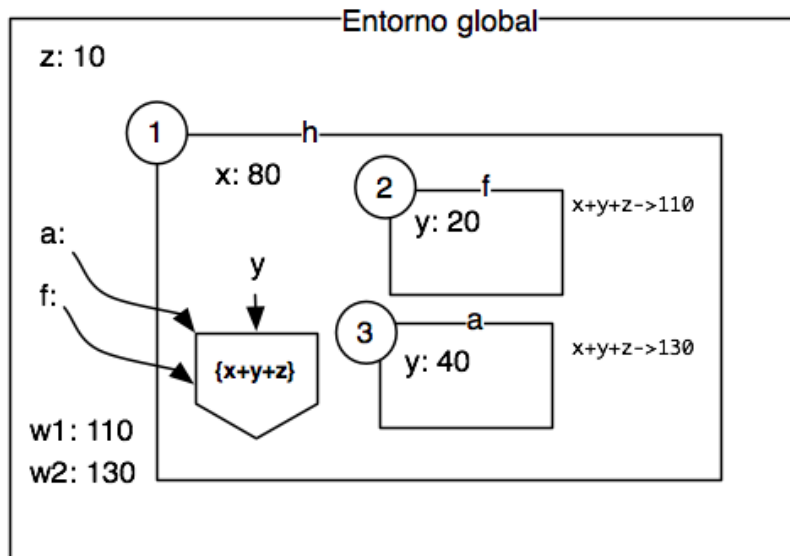
Ejercicio 6 (2 puntos)

El tema de clausuras con estado local lo dimos en Scala.

a) (1 punto) Dibuja el diagrama de ámbitos que se crea con la evaluación de las siguientes instrucciones en Scala:

```
val x = 10
val y = 20
def g(f:(Int,Int)=>Int) = {
  val x = 30
  val y = 40
  f(x,y)
}
val h = (x,y) => {x+y}
val a = g(h)
```

b) (1 punto) Dado el siguiente diagrama de ámbitos, escribe las instrucciones en Scala que lo crean:



Nombre: _____ DNI: _____

Lenguajes y Paradigmas de Programación

Curso 2013-2014

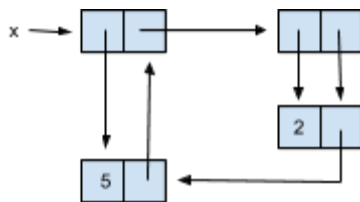
Tercer parcial

Normas importantes

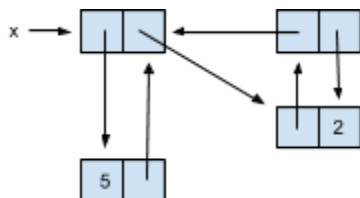
- La puntuación total del examen es de 10 puntos.
- Se debe contestar cada pregunta en las hojas que entregamos. Utiliza las últimas hojas para hacer pruebas. No olvides poner el nombre.
- La duración del examen es de 3 horas.

Ejercicio 1 (1 punto)

a) (0,5 punto) Escribe las instrucciones que generan el siguiente diagrama *box & pointer*:



b) (0,5 puntos) Dado el diagrama anterior, escribe las instrucciones que lo han modificado (utilizando como única referencia la x y sin añadir ningún valor ni pareja nueva) de la siguiente forma:



Ejercicio 2 (2 puntos)

Definimos en Scheme una estructura de datos formada por una lista enlazada con cabecera que contiene parejas con el dato y la posición que el dato tiene en la lista, sin elementos repetidos. Por ejemplo, la lista que contiene los datos 'a, 'z, 'c, 'd situados en la posición 1, 2, 3 y 4 se construiría de la siguiente forma:

```
(define lis (list '*cab* (cons 'a 1) (cons 'z 2) (cons 'c 3) (cons 'd 4)))  
⇒ (*cab* (a . 1) (z . 2) (c . 3) (d . 4))
```

Escribe la función mutadora (delete! x lista) que busca un elemento x en la lista y lo elimina de ella, mutando las posiciones de todos los que hay a continuación. Si el elemento no está en la lista, debe devolver el símbolo 'no-encontrado. Puedes utilizar o no la barrera de abstracción de las listas ordenadas con cabecera. Puedes implementar las funciones auxiliares que necesites.

```
(delete! 'z lis)  
lis  
⇒ (*cab* (a . 1) (c . 2) (d . 3))  
(delete! 'z lis)  
⇒ 'no-encontrado
```