Nombre:_____ **Grupo:**_____

Lenguajes y Paradigmas de Programación
Curso 2011-2012
Segundo 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 2 horas.**

Ejercicio 1 (0,75 punto)

Explica los siguientes conceptos:

a) **(0,25 puntos)** Barrera de abstracción
b) **(0,25 puntos)** Diferencia entre proceso recursivo y proceso iterativo
c) **(0,25 puntos)** *Currying*

Ejercicio 2 (1,75 puntos)

a) **(0,75 puntos)** Diseña e implementa en Scheme la barrera de abstracción del tipo de dato `rectángulo`, definido a partir de su vértice inferior izquierdo situado en la coordenada `x`,`y`, su base y su altura. Define constructores, selectores y al menos dos operadores.

b) **(0,75 puntos)** Implementa en Scheme la función `(engloba lista-rec)` que reciba una lista de rectángulos y devuelva el rectángulo que los englobe a todos.

c) **(0,75 puntos)** Explica si tu solución genera un proceso recursivo o iterativo.

Ejercicio 3 (1,5 puntos)

Define el procedimiento `(aplanar-f f exp-s)` que reciba una función unaria y una expresión-S como argumento y aplique la función f a todos los elementos de la exp-s y devuelva una lista plana con los resultados.

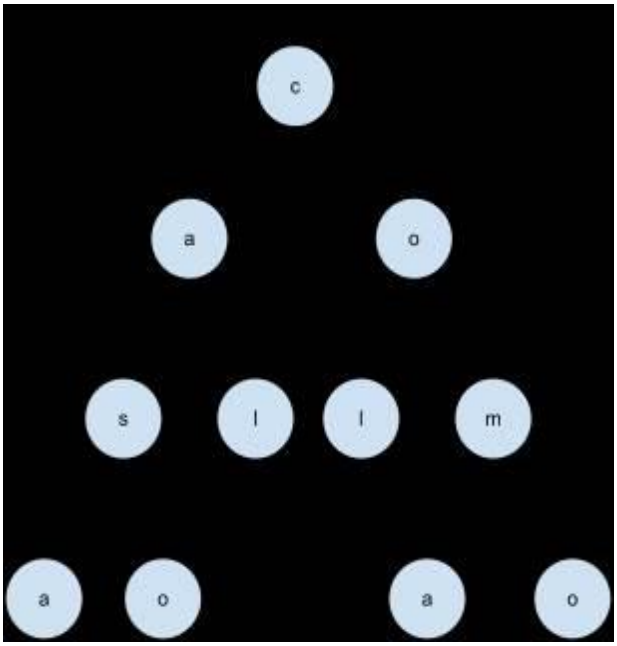
Ejemplo:

```
(aplanar-f (lambda (x) (* x x)) '(1 (2 (3) (4 (5 (6) 7))))
(1 4 9 16 25 36 49)
```

Ejercicio 4 (1,5 puntos)

a) **(0,5 puntos)** Define e implementa en Scheme la barrera de abstracción del árbol binario.

b) **(1 punto)** Define en Scheme el predicado `(palabra-b-tree? b-tree lista)` que reciba un árbol binario con letras y una lista que indique una palabra. Devuelve #t si la palabra se encuentra totalmente en una rama del árbol y #f en caso contrario.



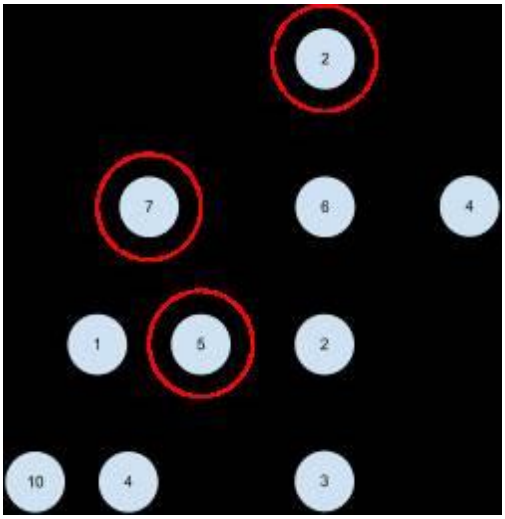

```
(palabra-b-tree b-tree '(c a s a)) → #t
(palabra-b-tree b-tree '(c a l l a)) → #f
(palabra-b-tree b-tree '(c o m)) → #f
```

Ejercicio 5 (1,5 puntos)

a) **(0,5 puntos)** Define e implementa en Scheme la barrera de abstracción del árbol genérico.

b) **(1 punto)** Define el procedimiento `(camino-max-tree tree)` que reciba un árbol genérico como argumento y devuelva una lista con los valores que se encuentra en un camino máximo: descendiendo siempre por la rama con el hijo que tiene un dato mayor.

Ejemplo:




```
(camino-max-tree tree) → (2 7 5)
```

Ejercicio 6 (1,5 puntos)

```
val x = 2
val y = 5
val z = 8
def h(z:Int) = {
   (x:Int) => x + y + z
}
def g(x:(Int)=>Int, y:Int, z:Int) = {
   x(y)
}
val f = h(3)
g(f,10,1)
```

a) **(1,25 puntos)** Dibuja y explica paso a paso cómo se crean los ámbitos generados tras la evaluación de las instrucciones anteriores en Scala.

b) **(0,25 puntos)** Indica el resultado que devuelve Scala

Ejercicio 7 (1,5 puntos)

a) **(0,75 puntos)** Define una función en Scala `creaBaraja()` que genere una lista de tuplas que representa la baraja española. Cada tupla será una tupla con un entero que representa el valor de la carta y una cadena que representa el palo. Los valores son enteros del 1 al 12 y los palos son "Oros", "Copas", "Bastos" y "Espadas". Define correctamente los tipos de los argumentos y del valor devuelto por la función.

Ejemplo:
`creaBaraja() → ((1,"Oros"),(2,"Oros"), … (11,"Espadas"),(12,"Espadas"))`

b) **(0,75 puntos)** Define la función recursiva filtraBaraja que reciba una baraja y un predicado de dos argumentos: un Int y un String. La función filtraBaraja devolverá una lista con las cartas que cumplan el predicado.

Ejemplo:

```
val baraja = creaBaraja()
def esOrosPar(s:String, y:Int) = {(s=="Oros") && (y % 2 == 0)}
filtraBaraja(esOrosPar _, baraja)  → ((2,"Oros"),...,(12,"Oros"))
```