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 puntos)

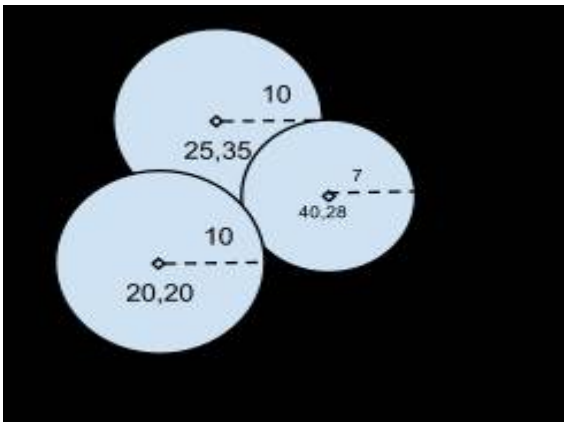
Explica los siguientes conceptos:

a) **(0,25 puntos)** *Tail recursion*
b) **(0,25 puntos)** *Memoization*
c) **(0,25 puntos)** Diferencia entre diseño e implementación de una barrera de abstracción

Ejercicio 2 (1,75 puntos)

a) **(0,75 puntos)** Diseña e implementa en Scheme la barrera de abstracción del tipo de dato `círculo`, definido a partir de un centro situado en la coordenada x,y y un radio. Define constructores, selectores y al menos dos operadores.

b) **(0,75 puntos)** Implementa en Scheme la función `(bounding-box lista-cir)` que reciba una lista de círculos y devuelva una lista con las cuatro coordenadas `(xmin,ymin,xmax,ymax)` del rectángulo que engloba a todos los círculos (*bounding-box*).



c) **(0,25 puntos)** Explica si tu solución genera un proceso recursivo o iterativo.

Ejercicio 3 (1,5 puntos)

Define en Scheme el procedimiento (`suma-exp-s exp-s1 exp-s2`) recursivo puro (sin usar *tail recursion*) que tome como parámetro dos expresiones-S de números con la misma estructura y devuelva una lista que contenga la suma de los elementos de ambas.

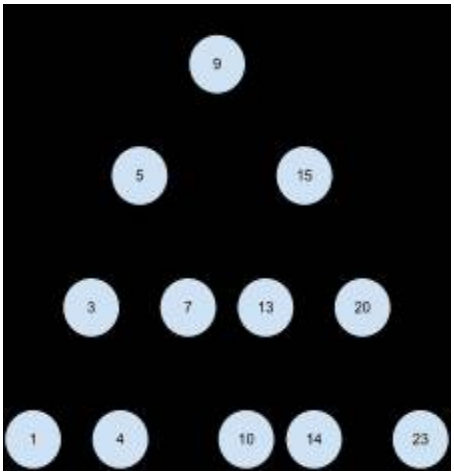
Ejemplo:

```
(suma-exp-s '(1 (2 (3) (4 (5 (6) 7)))) '(1 (3 (2) (1 (1 (1) 10)))))
(1 5 5 5 6 7 17)
```

Ejercicio 4 (1,5 puntos)

a) **(0,5 puntos)** Define e implementa en Scheme la barrera de abstracción de un árbol binario
b) **(1 punto)** Dado un árbol binario y un camino definido como una lista de símbolos: (< > = > > =) en el que:
<: indica que nos vamos por la rama izquierda
>: indica que nos vamos por la rama derecha
=: indica que nos quedamos con el dato de ese nodo.
Implementa la función `(camino-b-tree b-tree camino)` que devuelva una lista con los datos que indique el camino.

Ejemplo:

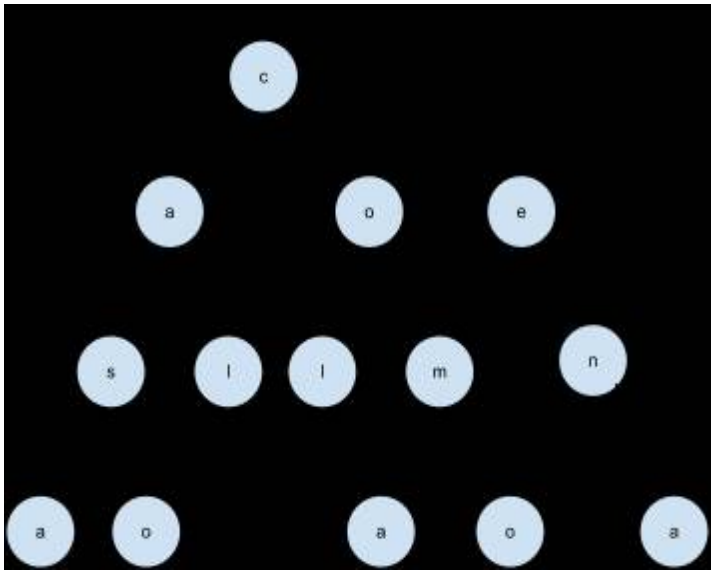



```
(camino-b-tree b-tree '(= < < = > =)) → (9 3 4)
(camino-b-tree b-tree '(> = < < =)) → (15 10)
```

Ejercicio 5 (1,5 puntos)

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

b) **(1 punto)** Define el predicado `(palabra-tree? tree lista)` que reciba un árbol genérico y una lista de caracteres como argumento. Devuelve #t si alguna rama del árbol coincide completamente con la palabra y #f en caso contrario.




```
(palabra-tree? tree '(c e n a)) → #t
(palabra-tree? tree '(c a s a s)) → #f
(palabra-tree? tree '(c o m)) → #f
```

Ejercicio 6 (1,5 puntos)

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

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 `parejasNums(x:Int)` que genere una lista con todas las tuplas de dos números desde 0 hasta x. Indica en la definición de la función el tipo que devuelve.

Ejemplo:
`parejasNums(2) → ((0,0),(0,1),(0,2),(1,0),(1,1),(1,2),(2,0),(2,1),(2,2))`

b) **(0,75 puntos)** Define la función recursiva `aplicaFuncionDosArgsLista` que tome una lista de tuplas de `Int` y una función que toma dos argumentos Int y devuelve un Int y que devuelva una lista con los enteros resultantes de aplicar la función a las tuplas. Define correctamente los tipos de los argumentos y del valor devuelto por la función.

Ejemplo:

```
val lista = parejasNums(2)
def suma(x:Int, y:Int) = x+y
aplicaFuncionDosArgsLista(suma _, lista)  → (0,1,2,1,2,3,2,3,4)
```