Nombre:

Lenguajes y Paradigmas de Programación
Curso 2010-2011
Segundo parcial, turno de mañana

Normas importantes

- La puntuación total del examen es de 20 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 1 hora.

Ejercicio 1 (5 puntos)

1) (3 puntos) Supongamos las siguiente definiciones en Scala:

```
class Persona
class Profesor extends Persona
class Estudiante extends Persona
def quienCorrige (lista:List[Profesor], a: Estudiante): Profesor = {...}
var pepe = new Estudiante
var juan = new Persona
var lista1 = List(new Profesor, new Estudiante)
var lista2 = List(new Profesor, new Profesor)
```

Indica cuales de las siguientes sentencias son incorrectas y explica por qué.

```
a) var p = quienCorrige(lista1, pepe)
b) var p = quienCorrige(lista2, pepe)
c) var p2: Persona = quienCorrige(lista2, pepe)
d) var p = quienCorrige(lista2, juan)
```

2) (2 puntos) Rellena los huecos con los tipos de datos correctos

```
def funcion1(a: Int, g: ____________): _______ = {
   var c: Int = g(a+1,a+2)
   c+2.0
}
```

Ejercicio 2 (5 puntos)

Define en Scala la función recursiva pura (sin recursión por la cola) `maxApply` que reciba dos parámetros: una lista `miLista` con un número par de enteros y una función cualquiera `g(x,y)`que se aplica a dos enteros y devuelve otro entero. La función `maxApply` debe aplicar la función `g` a los elementos de la lista `miLista` dos a dos y devolver el máximo.

Ejemplo:

```
def suma(x: Int, y: Int) = x+y
val miLista = List(2,3,5,-2,4,8)
maxApply (miLista, suma _)  → 12
```

Ejercicio 3 (5 puntos)

Dibuja y explica el diagrama de entornos de las siguientes instrucciones. ¿Qué valor devuelve la variable x?

```
var x=10
var y=20
def a() = {
   x=x+1
}
def b() = {
   var x = 50
   x = x+y
   var f = a _
   f
}
var g = b()
g()
x
```

Ejercicio 4 (5 puntos)

Implementa en Scheme el procedimiento mutador `aplana-expr!` que reciba una expresión-L (una lista con sublistas) y la transforme en una lista plana utilizando los mutadores `set-car!` y `set-cdr!`. Para simplificar, suponemos que el primer elemento de las listas y sublistas es atómico. Explica utilizando diagramas de caja y puntero el funcionamiento de tu solución.

Ejemplo:

```
(define lista1 '(1 2 3 (4 5) 6 7))
(define lista2 '(1 (2 (3 (4 5))) 6 7)
(aplana-expr! lista1)
(aplana-expr! lista2)
lista1 → (1 2 3 4 5 6 7)
lista2 → (1 2 3 4 5 6 7)
```