Nombre:

Lenguajes y Paradigmas de Programación
Curso 2010-2011
Primer parcial

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 (6 puntos)

Apartado 1 (1 punto)

a) Dibuja el diagrama box and pointer de la expresión:

```
(cons (cons 2 3) (cons (cons 1 2) (cons 3 (list (cons 5 6) 4 (cons 2 1)))))
```

b) ¿La expresión resultante es una lista? Explica tu respuesta.

Apartado 2 (2 puntos)

Representa en forma de árbol las siguientes expresiones:

Expresión-s: ((a b) (((c))) d)
Árbol binario: (23 (12 () (42 () ())) (8 (2 () ()) ()))
Árbol genérico: (a (b) (c (d (e)) (f)) (g (h)))

Apartado 3 (1 punto)

Empareja cada personaje con el lenguaje que diseñó. A continuación, ordena los lenguajes en orden cronológico:

Creadores: John Backus, Martin Odersky, John McCarthy, Yukihiro Matsumoto, Steel & Sussman

Creador	Lenguaje	Orden cronológico
	Ruby	
	Fortran	
	Scala	
	Scheme	
	Lisp	

Apartado 4 (2 puntos)

Explica en qué consisten los siguientes términos:

- Tail recursion
- Memoization
- Modelo de evaluación
- Ámbito

Ejercicio 2 (4 puntos)

Escribe el procedimiento `(num-tests lista-tests n)` que toma una lista de tests y un número n y que devuelva el número de tests de la lista que pasa el número n.

Por ejemplo, supongamos los tests `(mayor-que-8? x)`, `(par? x)`, `(impar? x)`

```
(define lista-tests (list mayor-que-8? par? impar?))
(num-tests lista-tests 12) -> 2
(num-tests lista-tests 3) -> 1
```

Ejercicio 3 (5 puntos)

Escribe un procemiento `(genera-tests lista-expresiones)` que devuelva una lista de tests a partir de una lista de expresiones. La lista de expresiones tendrá la forma de (op_1 n_1 … op_n n_n), donde los operadores pueden ser los símbolos >, < o =. Puedes definir funciones auxiliares.

Por ejemplo: `(genera-tests '(> 3 < 5 > 8 = 10))` devolverá una lista con 4 tests: la comprobación de si un número es mayor que 3, menor que 5, mayor que 8 y igual que 10. Esta lista se podría utilizar en el ejercicio anterior:

```
(num-tests (genera-tests '(> 3 < 5 > 8 = 10)) 10) -> 3
```

Ejercicio 4 (5 puntos)

a) Define la barrera de abstracción del árbol genérico y su implementación.

b) Utilizando esa barrera de abstracción, escribe el procedimiento `(mayor-num-hijos-tree tree)` que reciba un árbol genérico y devuelva número máximo de hijos que tenga un nodo del árbol.