Nombre:	 Grupo:
-	

Lenguajes y Paradigmas de Programación Curso 2013-2014

Primer parcial - Turno de mañana

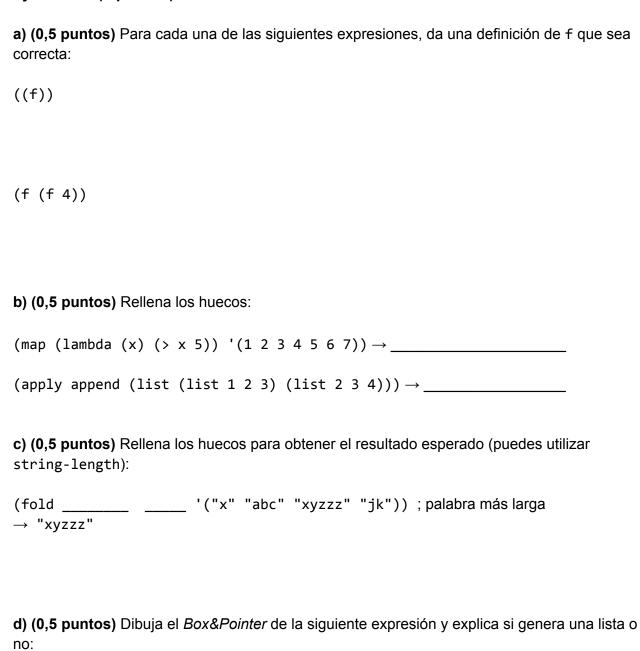
Normas importantes

- 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.

Ejercicio 1 (2 puntos)

- a) (0,75 puntos) Explica las funciones cons, car y cdr: qué parámetros aceptan y qué valores devuelven. Explica por qué el resultado de la expresión (cons 3 '(1 2 3 4)) es la lista '(3 1 2 3 4) y por qué el resultado de (cdr (cons 1 3)) es 3 en lugar de '(3). ¿Qué habría que cambiar en la expresión anterior para obtener '(3)?
- **b) (0,75 puntos)** Explica las características más importantes de cada uno de los siguientes paradigmas, indicando algún lenguaje de programación característico de cada uno de ellos:
 - Programación Funcional
 - o Programación Imperativa
 - Programación Orientada a Objetos
 - Programación Lógica
- c) (0,25 puntos) Indica el orden temporal de los lenguajes: Python, Java, Scheme, C
- d) (0,25 puntos) Indica el orden temporal de los siguientes hitos históricos relacionados con la historia de los computadores:
 - Primer programa avanzado de IA: juego de damas de Christopher Strachey
 - Arquitectura von Neumann
 - Máquina de Turing
 - UNIVAC

Ejercicio 2 (2 puntos)



(cons (cons 1 (cons 2 3)) (list (list 1 2) (cons 1 2)))

Ejercicio 3 (2 puntos)

a) (1 punto) Define la función recursiva (siguientes lista x) que reciba una lista y un elemento y devuelva una lista con los siguientes elementos de x en la lista. El elemento puede aparecer más de una vez.

Ejemplos:

```
(siguientes '(a b c d a c b c a) 'b) \rightarrow (c c) (siguientes '(a b c d a a c b a c a) 'a) \rightarrow (b a c c)
```

b) (1 punto) Define la función recursiva (min-max lista) que recibe una lista de números y devuelve una pareja con el mínimo y el máximo de la lista:

Ejemplos:

```
(\min-\max '(1 2 3 4 5 6)) \Rightarrow (1 . 6)
(\min-\max '(1 1 1 1 1) \Rightarrow (1 . 1)
```

Ejercicio 4 (2 puntos)

a) (0,75 puntos) Utilizando la función de orden superior que consideres más apropiada, define la función (suman-par lista-parejas) que reciba una lista de parejas de números y devuelva una lista con aquellas parejas cuya parte izquierda y derecha sumen un número par.

Ejemplo:

```
(suman-par '((1.2) (2.2) (3.6) (6.2) (4.5))) \rightarrow ((2.2)(6.2))
```

b) (1,25 punto) Define la función (aplica-n lista-funcs lista-orden n) que reciba dos listas, una de ellas contiene funciones unarias y la otra índices que referencian a posiciones de la primera lista (de 0 al número de elementos-1 de lista-funcs) y un número. Esta función deberá aplicar las funciones de lista-funcs al número n, en el orden indicado por lista-orden. Puedes utilizar list-ref, funciones auxiliares y/o de orden superior.

Ejemplo:

```
(aplica-n (list suma-1 mult-3 doble cuadrado) '(2 3 0 1 2 1) 3) \rightarrow 6050 ; (doble (cuadrado (suma-1 (mult-3 (doble (mult-3 3))))))
```

Ejercicio 5 (2 puntos)

Dados los siguientes fragmentos de código en Scheme, dibuja en un diagrama los ámbitos que se generan. Junto a cada ámbito escribe un número indicando en qué orden se ha creado. ¿Cuál es el resultado? ¿Cuántos ámbitos se crean? ¿Se crea alguna clausura?

a) (1 punto)

(g 14)

