Nombre:_____

Lenguajes y Paradigmas de Programación
Curso 2014-2015
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 (2,5 puntos)

a) (0,5 puntos) Dada la siguiente función en Scheme que define la serie de Fibonacci utilizando la técnica de *memoization*:

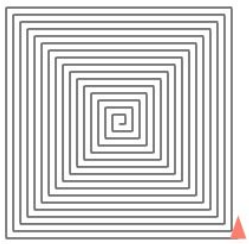
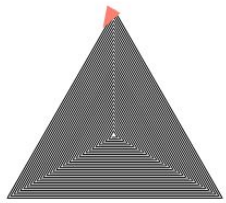
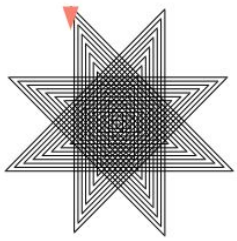
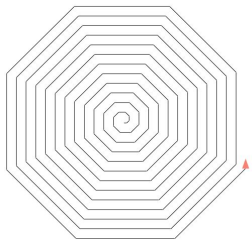
```
(define (fib-memo n lista)
    (cond ((= n 0) 0)
          ((= n 1) 1)
          ((not (null? (get n lista)))
           (put n lista))
          (else (let ((result
                       (+ (fib-memo (- n 1) lista)
                          (fib-memo (- n 2) lista))))
                  (begin
                    (put n lista)
                    result)))))
```

La función `fib-memo` contiene errores. Descríbelos y corrígelos.

b) (0,5 puntos) Dada la siguiente función en Scheme utilizando gráficos de tortuga:

```
(define (figura lado angulo)
   (if (< lado 200)
       (begin
         (draw lado)
         (turn angulo)
         (figura (+ lado 2) angulo))))
```

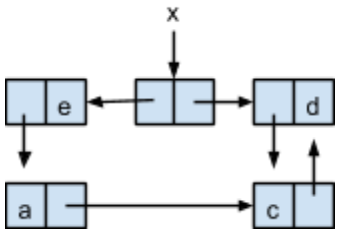
Indica qué se dibuja con la llamada `(figura 5 45)`

a) 	c) 
b) 	d) 

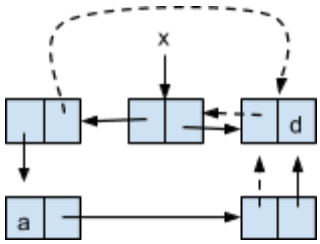
c) (0,5 puntos) Explica brevemente qué soluciones existen para mejorar el coste de la recursión. Ventaja e inconveniente de cada solución.

d) (0,5 puntos)

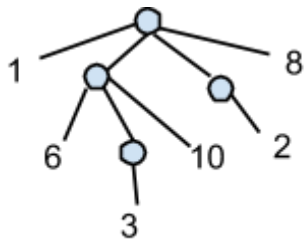
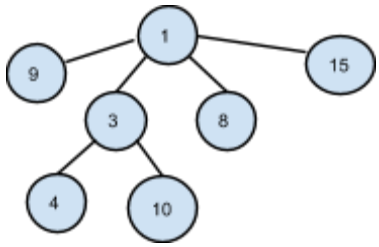
d.1) Escribe las instrucciones necesarias que generan el siguiente box & pointer:



d.2) Escribe las instrucciones necesarias que realizan los siguientes cambios (utilizando como única referencia la x)



e) (0,5 puntos) Dadas las siguientes estructuras de datos recursivas, escribe su expresión correspondiente en forma de lista estructurada y las instrucciones, utilizando la barrera de abstracción adecuada, para obtener el elemento 10 de cada una:

Pseudoárbol:  	Lista estructurada: (define lista '(_____)) Elemento 10:
Árbol:  	Lista estructurada: (define arbol '(_____)) Elemento 10:

Ejercicio 2 (1,5 puntos)

Escribe **utilizando recursión por la cola** la función `(cuadrado-lista lista)` que toma como argumento una lista de números y devuelve una lista con sus cuadrados.

Ejemplo:

```
(cuadrado-lista '(2 3 4 5)) ⇒ (4 9 16 25)
```

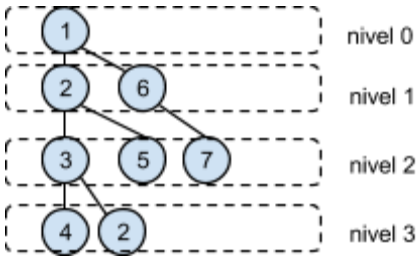
Ejercicio 3 (3 puntos)

a) (1,5 puntos) Escribe la función `(cuenta-diff-listas l1 l2)` que toma como argumentos dos listas estructuradas con la misma estructura pero con diferentes elementos y devuelve el número de elementos diferentes.

Ejemplos:

```
(cuenta-diff-listas '(a (b ((c)) d e) f) '(1 (b ((2)) 3 4) f)) ⇒ 4
(cuenta-diff-listas '((a b) c) '((a b) c)) ⇒ 0
```

b) (1,5 puntos) Escribe la función `(lista-nivel-tree nivel tree)` que reciba un nivel y un árbol y devuelva una lista con todos los nodos que se encuentran en ese nivel.




```
(lista-nivel-tree 0 tree) ⇒ (1)
(lista-nivel-tree 1 tree) ⇒ (2 6)
(lista-nivel-tree 2 tree) ⇒ (3 5 7)
(lista-nivel-tree 3 tree) ⇒ (4 2)
```

Ejercicio 4 (1,5 puntos)

Implementa el procedimiento mutador (`intercambia-elementos! lista`) que reciba una lista con cabecera con un número par de elementos e intercambie sus elementos de dos en dos. Debes proponer una solución que no utilice `set-car!`

Además **explica utilizando diagramas de caja y puntero** el funcionamiento de tu solución.

Ejemplos:

```
(define lista '(*clist* 1 2 3 4))
(intercambia-elementos! lista)
lista ⇒ (*clist* 2 1 4 3)
```