Lenguajes y Paradigmas de Programación
Curso 2004-2005
Examen de la Convocatoria de Septiembre

Normas importantes

- La puntuación total del examen es de 50 puntos que sumados a los 10 puntos de las prácticas dan el total de 60 puntos sobre los que se valora la nota de la asignatura.
- Para sumar los puntos de las prácticas **es necesario obtener un mínimo de 20 puntos en este examen**.
- Se debe contestar cada pregunta **en un hoja distinta**. No olvides poner el nombre en todas las hojas.
- La duración del examen es de 2 horas y 30 minutos.
- Las notas estarán disponibles en la web de la asignatura el próximo día 1 de Julio.

Pregunta 1 (10 puntos)
A) **(5 puntos)** Define una función `(cumplen-todos pred list)` que devuelva `#t` o `#f` dependiendo de si todos los elementos de la lista `list` cumplen el predicado `pred`. La lista `list` puede ser un pseudo-arbol y contener otras listas. Por ejemplo:

```
(cumplen-todos par? '(2 4 (6 8) 10)) -> #t
(cumplen-todos impar? '(1 3 5 6)) -> #f
(cumplen-todos impar? '()) -> #t
```

B) **(2 puntos)** La función que has definido en el apartado anterior ¿es iterativa o recursiva? ¿Por qué?

C) **(3 puntos)** Supongamos que uno de los elementos de la lista no cumple el predicado ¿se sigue procesando la lista en tu solución? En el caso de que no, perfecto, **ya puedes sumar los 3 puntos de este apartado**. En el caso de que se siga procesando la lista hasta el final, modifica la función para que no lo haga y devuelva `#f` justo en ese instante.

Pregunta 2 (10 puntos)
Vamos a crear un TAD para las horas del día (horas y minutos). Usaremos la representación de 24 horas, en que 3:00 PM es la hora 15:00. El constructor para las horas del día es `make-time`. Toma dos argumentos: las horas y los minutos. 4:12 debería ser `(make-time 4 12)` También queremos los siguientes tres operadores:

`(hour t)` → devuelve la parte de hora de un time dado
`(minute t)` → devuelve la parte de minutos de un time dado
`(hour? t)` → devuelve #t si el time t es una hora exacta (como 6:00); devuelve #f en cualquier otro caso

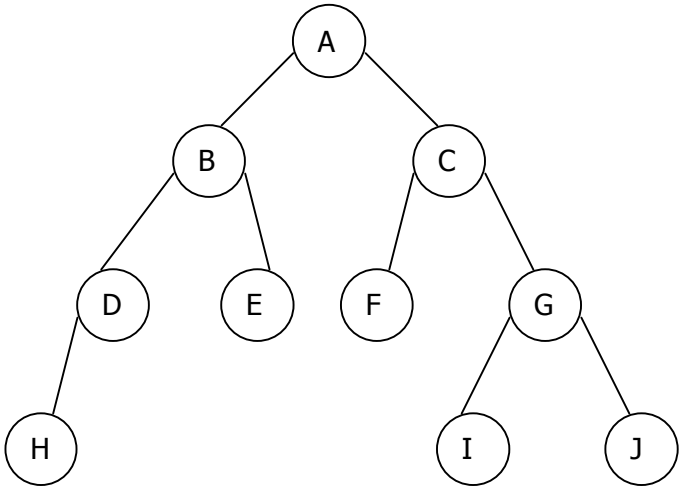
Ejemplo:

```
> (define t (make-time 4 12))
```

```
> (minute t)
12
> (hour t)
4
> (hour? t)
#f
```

Implementa el constructor `make-time` y los operadores `hour`, `minute` y `hour?` usando la técnica del paso de mensajes, en la que el objeto devuelto por el constructor es una función que admite mensajes.

Pregunta 3 (10 puntos)
El procedimiento `(prof_hojas tree)` devuelve una lista plana con la profundidad de cada hoja de un árbol binario. Ejemplo:



`(prof_hojas tree)` devolverá (3 2 2 3 3)

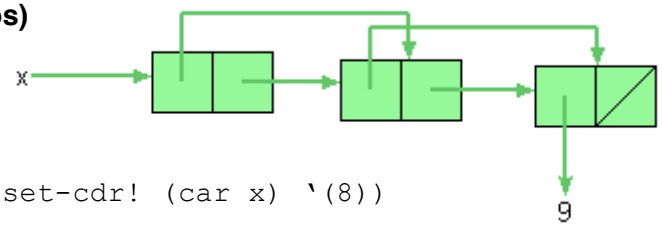
Rellena los huecos para que `prof_hojas` funcione correctamente:

```
(define (prof_hojas tree)
      (define (help tree depth)
      (cond ((null? tree) _____________________________)
      ((leaf? tree) _____________________________________)
      (else ( ___________________________________________
              ___________________________________________))))
      (help tree 0))
```

Pregunta 4 (10 puntos)
Para cada apartado:

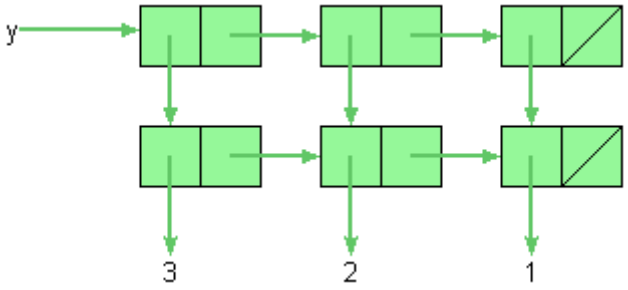
- Escribe una expresión Scheme que construya el diagrama caja y puntero de la figura.
- Dibuja el diagrama caja y puntero resultante de la mutación indicada.

A) **(5 puntos)**



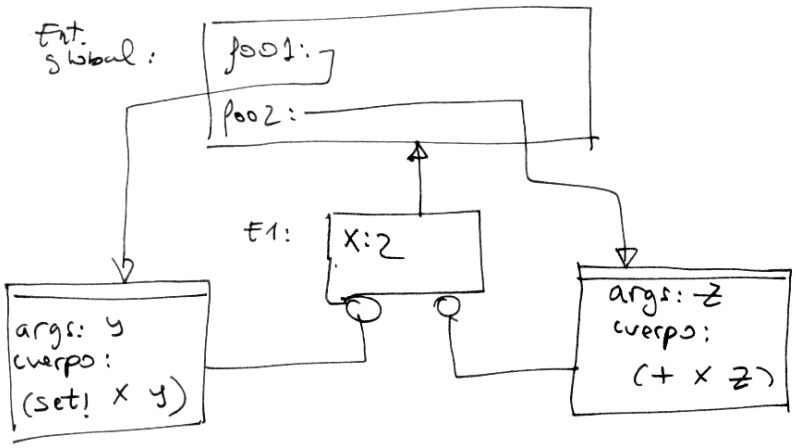
Mutación: `(set-cdr! (car x) '(8))`

B) **(5 puntos)**



Mutación: `(set-car! (cdr (cadr y)) 4)`

Pregunta 5 (10 puntos)
Supongamos el entorno definido por la siguiente figura



A) **(5 puntos)** Completa el siguiente programa para que genere el entorno anterior.

```
(define c  (________________)
(define foo1 (______))
(define foo2 (______))
```

Nota: La variable auxiliar `c` no aparece en la figura.

B) **(4 puntos)** Explica paso a paso cómo se han evaluado las expresiones del programa que acabas de completar para generar el entorno de la figura, según el modelo de evaluación de Scheme basado en entornos.

C) **(1 punto)** Escribe un ejemplo de invocación de `foo1` y `foo2`, indicando el resultado devuelto.