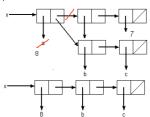
Soluciones Examen Junio 2007 LPP

```
Pregunta 1
```

1. b 2. a 3. b 4. c 5. a

Pregunta 2

a)



b)

'(8 b c)

Pregunta 3

a) (5 puntos)

Esto es una posible solución: hay bastantes más.

```
(let ((result (split (cdr lista) (- n 1))))
(cons
           (cons (car lista) (car result))
           (cdr result)))))
b) (5 puntos)
(define (do-split! lista n)
    result)
       (do-split! (cdr lista) (- n 1))))
(define (split! lista n)
   (if (equal? n 0)
(cons '() lista)
(cons lista (do-split! lista n))))
Pregunta 4
```

(define (tree-ref tree camino)
(cond ((null? tree) #f)
((null? camino) (dato tree))
(else (forest-ref (hijos tree) (car camino) (cdr

(define (forest-ref forest num-hijo camino)

(if (= num-hijo 1) (tree-ref (car forest) camino) (forest-ref (cdr forest) (- num-hijo 1) camino)))

Pregunta 5

camino)))))

a) Barrera de abstracción

Lo siguiente es una posible solución. Hay muchas, pero para ser correctas deben tener:

- Correcta especificación de las funciones (explicar cuáles son los argumentos y el resultado)
- Nombres correctos (incluir el sufijo con el tipo)
 Deben coincidir las funciones específicas de ambos tipos (la barrera de abstracción debe ser igual en ambos)

Funciones específicas del tipo "rect-ba" (rectángulo base altura):

(make-rect-ba x y base altura): devuelve un rectángulo con la posición (x y) y la base y altura especificada. (base-ba rect-ba): devuelve la base de un rectángulo-ba (altura-ba rect-ba): devuelve la altura de un rectángulo-ba (posx-ba rect-ba): devuelve la posición x de un rectángulo-ba (posy-ba rect-ba): devuelve la posición x de un rectángulo-ba

Funciones específicas del tipo "rect-coords" (rectángulo coordenadas):

(make-rect-coords x1 y1 x2 y2); devuelve un rectángulo con la esquina inf. izqd. situada en la posición (x1 y1) la esquina inf. dcha.en la posición (x2 y2) (base-coords rect-coords): devuelve la base de un rectángulo-coords (altura-coords rect-coords): devuelve la altura de un rectángulo-coords (posx-ba rect-coords): devuelve la posición x de un rectángulo-coords (posy-ba rect-coords): devuelve la posición y de un rectángulo-coords

(make-rect-from-base-altura x y base altura): devuelve un rectángulo con la posición (x y) y la base y altura especificada. (area rect): devuelve el área de un rectángulo (perimetro rect): devuelve el perimetro de un rectángulo

b) Implementación con paso de mensajes:

```
(define (make-rect-from-base-altura x y base altura)
```

```
cond
(((equal? mensaje 'base) base)
(((equal? mensaje 'altura) altura)
(((equal? mensaje 'x) x)
(((equal? mensaje 'y) y)
(else
                             (error "operacion desconocida")))))
(define (make-rect-from-coords x1 y1 x2 y2)
  (lambda (mensaje)
  (cond
    ((equal? mensaje 'base) (- x2 x1))
    ((equal? mensaje 'altura) (- y2 y1))
    ((equal? mensaje 'x) x1)
    ((equal? mensaje 'y) y2)
    (else
    (error "operacion desconcide")
                              (error "operacion desconocida")))))
Pregunta 6
Pregunta 7
a) 32
b)
                                                                                                     (define (g x)
(let ((z 8))
(lambda (y)
(set! z (+ x y))
...)))
(define h (g 4))
(h 6)
                                                    h:
                                  g:
                                                                       E1
                                                                                                     (let ((z 8))
(lambda (y) ...)
                                                                         x: 4/16
                                                                                                             (lambda (y)
(set! z (+ x y)
...)
        args: (x)
cuerpo:
(let ((z 8))
(lambda (y)
(set! z (+ x y))
...)
                                                                              E2 z: $ 10
                                                      args: (y)
cuerpo:
((set! z (+ x y)
...)
                                                                                                                        (set! z (+ x y))
(set! x (+ z y))
(+ x y z) -> 32
                                                                                                     y: 6
```

(lambda (mensaje) (cond

c) Sólo la función h tiene estado local y está representado por las variables x y z.