Ayudantía 11:
Preparación Quiz 4

Profesores: José Luis Martí Lara, Roberto Díaz Urra
Ayudantes:
Hugo Sepúlveda Arriaza
Gabriela Acuña Benito
Lucio Fondón Rebolledo
lucio.fondon@sansano.usm.cl

Universidad Técnica Federico Santa María
Departamento de Informática

Ejercicio 1

¿Cuál(es) de la(s) siguiente(s) afirmación(es) es(son) correcta(s)?

I. La evaluación de la expresión
`(map (lambda(x) (+ 1 (sqrt x))) '(9 16 25))` arroja un error.
II. La recursión simple puede ser más fácil de implementar, pero menos eficiente que la recursión de cola.
III. Se puede hacer un número indefinido de llamados de recursión de cola sin causar overflow del stack.
IV. Un string se escribe usando comillas simples.

a) I y II
b) II y III
c) II, III y IV
d) I y III

Ejercicio 1

¿Cuál(es) de la(s) siguiente(s) afirmación(es) es(son) correcta(s)?

I. La evaluación de la expresión
`(map (lambda(x) (+ 1 (sqrt x))) '(9 16 25))` arroja un error.
II. La recursión simple puede ser más fácil de implementar, pero menos eficiente que la recursión de cola.
III. Se puede hacer un número indefinido de llamados de recursión de cola sin causar overflow del stack.
IV. Un string se escribe usando comillas simples.

a) I y II
b) II y III
c) II, III y IV
d) I y III

R: **Alternativa b**

Ejercicio 2

¿Cuál es la salida de la siguiente consulta?:

```
>'(cons 'a (cons 'b '(c d (append '(e f) '(g h)))))
```

a) `(cons 'a (cons 'b '(c d (append '(e f) '(g h)))))`
b) `(a (b (c d (e f g h))))`
c) `(a b c d e f g h)`.
d) `null`.

Ejercicio 2

¿Cuál es la salida de la siguiente consulta?:

```
>'(cons 'a (cons 'b '(c d (append '(e f) '(g h)))))
```

a) `(cons 'a (cons 'b '(c d (append '(e f) '(g h)))))`
b) `(a (b (c d (e f g h))))`
c) `(a b c d e f g h)`.
d) `null`.

R: **Alternativa a**

Ejercicio 3

¿Cuál de las siguientes funciones es la única con recursividad de cola?

```
;; a)
(define f1
    (lambda (lista)
        (cond ((null? lista) 0)
        (else (+ (car lista) (f1 (cdr lista)))))))
```

```
;; b)
(define f2
    (lambda (lista)
        (cond ((null? lista) 0)
            (else (f2 (cdr lista))))))
```

```
;;c)
(define f3
    (lambda (lista)
        (cond ((null? lista) 0)
            ((= (length lista) 1) (f3 (cdr lista)))
            (else (+ (car lista) (f3 (cdr lista)))))))
```

```
;;d)
(define f4
    (lambda (lista)
        (cond ((null? lista) '())
        (else (cons (car lista)(f4 (cdr lista)))))))
```

Ejercicio 3

¿Cuál de las siguientes funciones es la única con recursividad de cola?

```
;; a)
(define f1
    (lambda (lista)
        (cond ((null? lista) 0)
        (else (+ (car lista) (f1 (cdr lista)))))))
```

```
;; b)
(define f2
    (lambda (lista)
        (cond ((null? lista) 0)
            (else (f2 (cdr lista))))))
```

```
;;c)
(define f3
    (lambda (lista)
        (cond ((null? lista) 0)
            ((= (length lista) 1) (f3 (cdr lista)))
            (else (+ (car lista) (f3 (cdr lista)))))))
```

```
;;d)
(define f4
    (lambda (lista)
        (cond ((null? lista) '())
        (else (cons (car lista)(f4 (cdr lista)))))))
```

R: **Alternativa b**

Ejercicio 4

¿Cuál es la salida de la siguiente consulta?:

```
>(cdr (car (cdr (cdr (cdr '((1 2) 3 4 (5 6 7))))))
```

a) ()
b) (4)
c) (6 7)
d) ((5 6 7))

Ejercicio 4

¿Cuál es la salida de la siguiente consulta?:

```
>(cdr (car (cdr (cdr (cdr '((1 2) 3 4 (5 6 7))))))
```

a) ()
b) (4)
c) (6 7)
d) ((5 6 7))

R: **Alternativa c**

Ejercicio 5

En relación a las listas en scheme es falso que:

a) Las listas contemplan elementos de cualquier tipo.
b) Una función se escribe como una lista, donde el primer elemento es la función y los siguientes los parámetros.
c) Para que una lista sea evaluada se le antecede con una citación simple.
d) Las funciones `car` y `cdr` se utilizan para obtener el primer elemento y una lista con el resto de los elementos respectivamente.

Ejercicio 5

En relación a las listas en scheme es falso que:

a) Las listas contemplan elementos de cualquier tipo.
b) Una función se escribe como una lista, donde el primer elemento es la función y los siguientes los parámetros.
c) Para que una lista sea evaluada se le antecede con una citación simple.
d) Las funciones `car` y `cdr` se utilizan para obtener el primer elemento y una lista con el resto de los elementos respectivamente.

R: **Alternativa c**

Ejercicio 6

¿Cuál de las expresiones no es equivalente a la siguiente expresión?:

```
>(define proc (lambda (x) (* x x)))
```

I. `(define proc (lambda (x) (let ((y x)) (* y y))))`
II. `(define (proc x) (let ((y x)) (* y y)))`
III. `(define (proc x) (* x x))`

a) I
b) II
c) III
d) Todas son equivalentes

Ejercicio 6

¿Cuál de las expresiones no es equivalente a la siguiente expresión?:

```
>(define proc (lambda (x) (* x x)))
```

I. `(define proc (lambda (x) (let ((y x)) (* y y))))`
II. `(define (proc x) (let ((y x)) (* y y)))`
III. `(define (proc x) (* x x))`

a) I
b) II
c) III
d) Todas son equivalentes

R: **Alternativa d**

Ejercicio 7

En relación a Scheme, es cierto que:

I. Una expresión `lambda` es un objeto tipo procedimiento que no tiene nombre.
II. La evaluación de funciones está controlada por recursión e iteración.
III. La recursividad de cola nos permite ahorrar memoria de stack.
IV. `(equal? '(1 2 3) (1 2 3))` retorna `#t`

a) I y III
b) I, II y III
c) II, III y IV
d) Todas las anteriores

Ejercicio 7

En relación a Scheme, es cierto que:

I. Una expresión `lambda` es un objeto tipo procedimiento que no tiene nombre.
II. La evaluación de funciones está controlada por recursión e iteración.
III. La recursividad de cola nos permite ahorrar memoria de stack.
IV. `(equal? '(1 2 3) (1 2 3))` retorna `#t`

a) I y III
b) I, II y III
c) II, III y IV
d) Todas las anteriores

R: **Alternativa a**