

# Control 1 – Lenguajes de Programación

Departamento de Ciencias de la Computación

Universidad de Chile

Profesor: Éric Tanter

13 de Abril del 2008

2 horas / 1.5 puntos por pregunta
-----------------------------------

1. Mergesort en Scheme (no olvidar especificar contratos, tests, etc.)

- Implemente en Scheme la función **merge** que une dos listas ordenadas en una nueva lista ordenada. Esta función es parametrizada por una operación de comparación:

```
(merge '(1 5 9) '(2 6 7 10) <) -> (1 2 5 6 7 9 10)
(merge '(9 5 1) '(10 7 6 2) >) -> (10 9 7 6 5 2 1)
```

Note que ambas listas podrían no tener el mismo tamaño.

- Defina ahora la función **merge-sort**, que use la función anterior, y sea

```
(merge-sort '(10 5 2 6 7 1) <)
-> (1 2 5 6 7 10)
```

**Pauta**, ver archivo **merge-sort.scm**

2. La representación de un árbol binario esta dado por la definición

```
bintree := ()
          | (<symbol> . <bintree> . <bintree>)
```

Donde un árbol binario puede ser o vacío o un nodo. El nodo de un árbol consta de la raíz (un símbolo Scheme), una rama izquierda y una rama derecha.

Implementa la función (**reverse tree**). Retornando como resultado un nuevo árbol similar a **tree**, pero donde en cada nodo las ramas han sido intercambiadas (rama izquierda a la derecha y viceversa). Por ej.:

```

(reverse '(8 (13 () (12 () ()))
          (20 () ())))
->
'(8 (20 () ())
  (13 (12 () ()) ()))

```

**Pauta,** ver archivo `bintree-reverse.scm`

3. Considere un lenguaje con la siguiente gramática:

```

<expr> ::= <id> | <num>
         | (+ <expr> <expr>)
         | (if <expr> <expr> <expr>)
         | (lambda (<id>) <body>)
         | (<expr> <expr>)

```

Defina en Scheme la función `free-vars` que retorna la lista de variables libres de una expresión dada. No se olvide de partir escribiendo al menos un test por cada tipo de expresión.

**Pauta,** ver archivo `free-vars.scm`

4. Considere el lenguaje WAE extendido con una expresión para imprimir valores en la pantalla (cuyo valor es no especificado), y una expresión para hacer una secuencia de dos expresiones (cuyo valor es el valor de la última expresión).

```

<WAE> ::= <id> | <num>
         | {+ <WAE> <WAE>}
         | {- <WAE> <WAE>}
         | {with {<id> <WAE>} <WAE>}
         | {print <WAE>} // retorna unspecified
         | {seq <WAE> <WAE>} // retorna el valor de la 2nda expr

```

**Nota:** Los pasos de evaluación descritos a continuación solo engloban: la evaluación, la substitución y el estado de la pantalla con cada impresión.

- (0,5 pts) Describa los pasos de evaluación de la siguiente expresión, usando un régimen de substitución temprana:

```

{with {x {+ 3 4}}
  {with {y {seq {print x}
                {+ x 1}}}}
  {+ y y}}}

```

Para cada paso, precisa si se trata de un paso de evaluación o de substitución. También mantenga actualizado el estado de la pantalla.

**Pauta:**

1. Evaluación: `{+ 3 4}` resultado: 7
2. Substitución: `'x` por: 7 en: `{print x}`
3. Estado de Pantalla: 7
4. Substitución: `'x` por: 7 en: `{+ x 1}`
5. Evaluación: `{+ 7 1}` resultado: 8
6. Substitución: `'y` por: 8 en: `{+ y y}`
7. Evaluación: `{+ 8 8}` resultado: 16

- (0,5 pts) Ahora, reduzca la misma expresión, usando un régimen de substitución perezosa.

**Pauta:**

1. Substitución: `'y` por: `{seq {print x} {+ x 1}}` en: `{+ y y}`  
 -- 1ra ocurrencia de `'y` en `{+ y y}`
2. Substitución: `'x` por: `{+ 3 4}` en: `{print x}`
3. Evaluación: `{+ 3 4}` resultado: 7
4. Substitución de Pantalla: 7
5. Substitución: `'x` por: `{+ 3 4}` en: `{+ x 1}`
6. Evaluación: `{+ {+ 3 4} 1}` resultado: 8  
 -- 2da ocurrencia de `'y` en `{+ y y}`
7. Substitución: `'x` por: `{+ 3 4}` en: `{print x}`
8. Evaluación: `{+ 3 4}` resultado: 7
9. Estado de Pantalla: 7 7
10. Substitución: `'x` por: `{+ 3 4}` en: `{+ x 1}`
11. Evaluación: `{+ {+ 3 4} 1}` resultado: 8
12. Evaluación: `{+ 8 8}` resultado: 16

- (0,5 pts) ¿son equivalentes ambos regímenes? ¿por qué?

**Pauta:** Ambos regímenes no son equivalentes apesar de retornar el mismo resultado. Esto se debe, en el ejemplo, a la diferencia en el estado de pantalla. La substitución eager imprimirá una sola vez el valor de 7, mientras que la substitución lazy lo imprimirá dos veces.