# Auxiliar Cuatro - CC41A
# **Lenguajes de Programación**
## *Substituci'on y Funciones (1er Parte)*

Oscar E. A. Callaú
*oalvarez@dcc.uchile.cl*

Santiago - Chile, 25/Ago/2008

## 1. Substituación Explícita, C1 I-2007

Considere la siguiente implementación de la función de substitución `subst` del lenguaje WAE visto en clases:

```
;; subst: WAE symbol WAE -> WAE
(define (subst expr sub-id val)
  (type-case WAE expr
    (num (n) expr)
    (add (l r) (add (subst l sub-id val)
                    (subst r sub-id val)))
    (sub (l r) (sub (subst l sub-id val)
                    (subst r sub-id val)))
    (with (bound-id named-expr bound-body)
          (if (symbol=? bound-id sub-id)
              expr
              (with bound-id
                    named-expr
                    (subst bound-body sub-id val))))
    (id (v) (if (symbol=? v sub-id) val expr))))
```

Con esta función de substitución, el calculador se cae en las dos expresiones siguientes:

1. `{with {x 5} {with {y x} y}}`
2. `{with {x 5} {with {x x} x}}`

Para cada caso explique porqué se cae el calculador, y luego de una versión corregida de `subst`.

## 2. Más Substitución

En que casos y porque falla esta implementación de substitución

```
(define (subst expr sub-id val)
  (type-case WAE expr
             [num (n) expr]
             [add (l r) (add (subst l sub-id val) (subst r sub-id val))]
             [sub (l r) (sub (subst l sub-id val) (subst r sub-id val))]
             [with (bound-id named-expr bound-body)
                   (if (symbol=? bound-id sub-id)
                       (with bound-id
                             named-expr
                             (subst bound-body sub-id val))
                       (with bound-id
                             (subst named-expr sub-id val)
                             (subst bound-body sub-id val)))]
             [id (v) (if (symbol=? v sub-id) val expr)]))
```

## 3. Cálculo Lambda

Según la siguiente definición de $\Lambda$-cálculo, en Gramática Concreta:

```
(define-type Lambda-Calculus
  (Id (id symbol?))
  (Lambda (parameter symbol?) (body Lambda-Calculus?))
  (App (function Lambda-Calculus) (argument Lambda-Calculus)))
```

implemente la función:

**eqLambda** , el cual toma dos expresiones $\lambda$ como parámetros y retorna si ellas son **exactamente** iguales.