1. Realizar las siguientes actividades sobre Cálculo λ.

   (a) Da una expresión del Cálculo λ que involucre todos los constructores del lenguaje (variables, abstracciones λ y aplicaciones de función).
   Solución. $(\lambda x.x)(\lambda y.y)y$

   (b) Para cada una de las siguientes expresiones:
   (a) Currificar en caso de ser necesario.
   (b) Dar una expresión que sea α-equivalente.
   (c) Indicar las variables de ligado, ligadas y libres.
   (d) Realizar las β-reducciones correspondientes hasta llegar a su Forma Normal o indicar por qué ésta no existe.

   a) $(\lambda abc.cba)zz(\lambda wv.w)$
   Solución.
   (a) $(\lambda a.\lambda b.\lambda c.cba)zz(\lambda w.\lambda v.w)$
   (b) $(\lambda a.\lambda b.\lambda c.cba)zz(\lambda w.\lambda v.w)[a := x]$
   $\equiv_\alpha (\lambda x.\lambda b.\lambda c.cbx)zz(\lambda w.\lambda v.w)[b := y]$
   $\equiv_\alpha (\lambda x.\lambda y.\lambda c.cyx)zz(\lambda w.\lambda v.w)[c := z]$
   $\equiv_\alpha (\lambda x.\lambda y.\lambda z.zyx)zz(\lambda w.\lambda v.w)[w := m]$
   $\equiv_\alpha (\lambda x.\lambda y.\lambda z.zyx)zz(\lambda m.\lambda v.m)[v := n]$
   $\equiv_\alpha (\lambda x.\lambda y.\lambda z.zyx)zz(\lambda m.\lambda n.m)$
   (c) $(\lambda \textcolor{red}{a}.\lambda \textcolor{red}{b}.\lambda \textcolor{red}{c}.\textcolor{green}{cba})\textcolor{blue}{zz}(\lambda \textcolor{red}{w}.\lambda \textcolor{red}{v}.\textcolor{green}{w})$
   (d) $(\lambda a.\lambda b.\lambda c.cba)zz(\lambda w.\lambda v.w) \to_\beta \lambda b.\lambda c.cba[a := zz(\lambda w.\lambda v.w)]$
   $= \lambda b.\lambda c.cbzz(\lambda w.\lambda v.w) \to_\beta \lambda c.cbzz[b := (\lambda w.\lambda v.w)]$
   $= \lambda c.c(\lambda w.\lambda v.w)zz \to_\beta c[c := (\lambda w.\lambda v.w)zz]$
   $= (\lambda w.\lambda v.w)zz \to_\beta \lambda v.w[w := zz]$
   $= \lambda v.zz$

   b) $(\lambda y.y)(\lambda x.xx)(\lambda z.zq)$
   Solución.
   (a) No es necesario currificar.
   (b) $(\lambda y.y)(\lambda x.xx)(\lambda z.zq)[x := a]$
   $\equiv_\alpha (\lambda y.y)(\lambda a.aa)(\lambda z.zq)[y := b]$
   $\equiv_\alpha (\lambda b.b)(\lambda a.aa)(\lambda z.zq)[z := c]$
   $\equiv_\alpha (\lambda b.b)(\lambda a.aa)(\lambda c.cq)$
   (c) $(\lambda \textcolor{red}{y}.\textcolor{green}{y})(\lambda \textcolor{red}{x}.\textcolor{green}{xx})(\lambda \textcolor{red}{z}.\textcolor{green}{z}\textcolor{blue}{q})$
   (d) $(\lambda y.y)(\lambda x.xx)(\lambda z.zq) \to_\beta y[y := (\lambda x.xx)(\lambda z.zq)]$
   $= (\lambda x.xx)(\lambda z.zq) \to_\beta xx[x := (\lambda z.zq)]$
   $= (\lambda z.zq)(\lambda z.zq) \to_\beta zq[z := (\lambda z.zq)]$
   $= (\lambda z.zq)q \to_\beta zq[z := q]$
   $= qq$

   c) $(\lambda z.z)(\lambda z.zz)(\lambda z.zy)$
   Solución.
   (a) No es necesario currificar.
   (b) $(\lambda z.z)(\lambda z.zz)(\lambda z.zy)[z := a]$
   $\equiv_\alpha (\lambda a.a)(\lambda a.aa)(\lambda a.ay)$
   (c) $(\lambda \textcolor{red}{z}.\textcolor{green}{z})(\lambda \textcolor{red}{z}.\textcolor{green}{zz})(\lambda \textcolor{red}{z}.\textcolor{green}{z}\textcolor{blue}{y})$

(d) $(\lambda z.z)(\lambda z.zz)(\lambda z.zy) \to_\beta z[z := (\lambda z.zz)(\lambda z.zy)]$
$= (\lambda z.zz)(\lambda z.zy) \to_\beta zz[z := (\lambda z.zy)]$
$= (\lambda z.zy)(\lambda z.zy) \to_\beta zy[z := (\lambda z.zy)]$
$= (\lambda z.zy)y \to_\beta zy[z := y]$
$= yy$

d) $(\lambda a.aa)(\lambda b.ba)c$
Solución.

(a) No es necesario currificar.

(b) $(\lambda a.aa)(\lambda b.ba)c\ [a := x]$
$\equiv_\alpha (\lambda x.xx)(\lambda b.bx)c\ [b := y]$
$\equiv_\alpha (\lambda x.xx)(\lambda y.yx)c$

(c) $(\lambda a.aa)(\lambda b.ba)c$

(d) $(\lambda a.aa)(\lambda b.ba)c \to_\beta aa[a := (\lambda b.ba)c]$
$= (\lambda b.ba)c(\lambda b.ba)c \to_\beta ba[b := c(\lambda b.ba)c]$
$= c(\lambda b.ba)ca \to_\beta ba[b := ca]$
$= ccaca$

2. De acuerdo a la representación de números (Numerales de Church) y representación de booleanos en el Cálculo λ.

(a) Definir la función $\neq$ que decide si un número es distinto a otro.
Solución. $\neq =_{def.} \lambda xy.if\ x == y\ Then\ \lambda t.\lambda f.t\ else\ \lambda t.\lambda f.f$

(b) Definir la función $\to$ (implicación) sobre booleanos.
Solución. $\to =_{def.} \lambda ab.if\ a\ Then\ b\ else\ \lambda t.\lambda f.t$

(c) Definir la función *nor* sobre booleanos.
Solución. $nor =_{def.} \lambda p.\lambda q.pqq$
Veamos el ejemplo $nor\ True\ False$.
Dado que $True =_{def.} \lambda t.\lambda f.t$ y $False =_{def.} \lambda t.\lambda f.f$ entonces.
$nor\ True\ False = (\lambda p.\lambda q.pqq)(\lambda t.\lambda f.t)(\lambda t.\lambda f.f)$
$= (\lambda t.\lambda f.t)(\lambda t.\lambda f.f)(\lambda t.\lambda f.f)$
$= \lambda t.\lambda f.f \equiv False.$

3. Explica por qué el algoritmo de sustitución visto en clase es de $O(n^2)$.
Solución. Es de tal complejidad ya que tenemos expresiones anidadas con n variables, es decir, de forma general si tenemos n variables, tenemos lo siguiente.

$$n + (n-1) + (n-2) + ... + 3 + 2 + 1 = \frac{n(n-1)}{2}$$

Razón por la cual tenemos complejidad $O(n^2)$.

4. Evalúa las siguientes expresiones usando:
   (a) Sustitución.
   (b) Ambientes con alcance dinámico.
   (c) Ambientes con alcance estático.
   Para (a) y (b) es necesario que muestres el ambiente de evaluación en forma de pila.

   (a)
```
(let (a 2)
  (let (b 3)
    (let (foo (lambda (x) (- (+ a b) x)))
      (let (a -2)
        (let (b -3)
          (let (foo (lambda (x) (+ (- a b) x)))
            (foo -10)))))))
```

Solución.

a) Sustitución.

(let (a 2) [a:=2]
  (let (b 3) [a:=2]
    (let (foo (lambda (x) (+ (- a[a:=2] b) x))) [a:=2]
      (let (b -3) [a:=2]
        (let (foo (lambda (x) (+ (- a[a:=2] b) x))) [a:=2]
          (foo -10)))))) [a:=2]

(let (a 2)
  (let (b 3) [b:=3]
    (let (foo (lambda (x) (+ (- 2 b[b:=3]) x))) [b:=3]
      (let (b -3) [b:=3]
        (let (foo (lambda (x) (+ (- 2 b[b:=3]) x))) [b:=3]
          (foo -10)))))) [b:=3]

(let (a 2)
  (let (b 3)
    (let (foo (lambda (x) (+ (- 2 3) x))) [foo:=(+ (- 2 3) x)]
      (let (b -3) [foo:=(+ (- 2 3) x)]
        (let (foo (lambda (x) (+ (- 2 3) x))) [foo:=(+ (- 2 3) x)]
          (foo -10)))))) [foo:=(+ (- 2 3) x)]

(let (a 2)
  (let (b 3)
    (let (+ -1 x) x)) [foo:=(+ (- 2 3) x)]
      (let (b -3) [foo:=(+ (- 2 3) x)]
        (let (foo (lambda (x) (+ (- 2 3) x))) [foo:=(+ (- 2 3) x)]
          ((+ (- 2 3) x) -10)))) [foo:=(+ (- 2 3) x)]

(let (a 2)
  (let (b 3)

```
        (let (+ -1 x) x))
            (let (b -3)
                (let (foo (lambda (x) (+ (- 2 3) x)))
                    ((+ -1 -10) -10))))
```

```
(let (a 2)
    (let (b 3)
        (let (+ -1 x) x))
            (let (b -3)
                (let (foo (lambda (x) (+ (- 2 3) x)))
                    (-1))))
```

b) Dinámico.
Primero hacemos la pila con los valores establecidos en los *let* quedando de la siguiente manera.

x	-10
b	-3
a	-2
foo	((lambda (x) (- (+ a b) x)) -10)
b	3
a	2

A continuación tomamos la función que está en medio de la expresión.
→ (foo -10)
→ ((lambda (x) (- (+ a b) x)) -10)
Trabajamos ahora sobre el cuerpo de la función.
→ (- (+ a b) x)
→ (- (+ (-2) (-3)) -10)
→ (- -5 -10)
→ 5
Por lo tanto, la evaluación de la expresión en ambiente dinámico es 5.

c) Estático.
Primero hacemos la pila con los valores establecidos en los *let* quedando de la siguiente manera.

x	-10
~~b~~	~~-3~~
~~a~~	~~-2~~
~~foo~~	~~((lambda (x) (- (+ a b) x)) -10)~~
b	3
a	2

A continuación tomamos la función que está en medio de la expresión y hacemos la cerradura.

→ (foo -10)
→ (<x, (- (+ a b) x), (a 2)(b 3)> -10)
Trabajamos ahora sobre el cuerpo de la cerradura.
→ (- (+ a b) x)
→ (- (+ 2 3) -10)
→ (- 5 -10)
→ 15
Por lo tanto, la evaluación de la expresión en ambiente estático es 15.

(b)
```
(let (foo (lambda (x) (+ x (foo (- x 1)))))
  (foo 10))
```

Solución.

a) Sustitución.
(let (foo (lambda (x) (+ x (foo (- x 1))))) [foo:=(+ x (foo (- x 1)))]
(foo 10)) [foo:=(+ x (foo (- x 1)))]

(let (+ x (+ x (foo (- x 1)))) [foo:=(- x 1)]
((+ x (foo (- x 1))) 10)) [foo:=(- x 1)]

(let (+ x (- x 1)) [x:=10]
((+ x (- x 1)) 10)) [x:=10]

(let (+ x (- 10 1))
((+ x (- 10 1)) 10))

(let (+ x 9)
((+ 9 9) 10))

(let (+ x 9)
(+ 18 10))

(let (+ x 9)
28)

b) Dinámico.
Primero hacemos la pila con los valores establecidos en los *let* quedando de la siguiente manera.

x	10
foo	(lambda (x) (+ x (foo (- x 1))))

A continuación tomamos la función que está en medio de la expresión.
→ (foo 10)
→ ((lambda (x) (+ x (foo (- x 1)))) 10)

Trabajamos ahora sobre el cuerpo de la función.
→ (+ x (foo (- x 1)))
→ (+ 10 (foo (- 10 1)))
→ (+ 10 (foo 9)))
Como la función tiene otra aparición dentro, entonces la función que está adentro se va a ciclar provocando que no se le pueda dar algún valor.

c) Estático.
Primero hacemos la pila con los valores establecidos en los *let* quedando de la siguiente manera.

x	10
foo	(lambda (x) (+ x (foo (- x 1))))

A continuación tomamos la función que está en medio de la expresión y hacemos la cerradura.
→ (foo 10)
→ (<x, (+ x (foo (- x 1))), (x 10)> 10)
Trabajamos ahora sobre el cuerpo de la cerradura.
→ (+ x (foo (- x 1)))
→ (+ 10 (foo (- 10 1)))
→ (+ 10 (foo 9)))
Como la función tiene otra aparición dentro, entonces la función que está adentro se va a ciclar provocando que no se le pueda dar algún valor.

(c)
```
(let (x 2)
    (let (foo (lambda (a) (+ x 2)))
        (let (y 3)
            (let (foo (lambda (b) (- y b)))
                (let (x 4)
                    (let (goo (lambda (b) (+ (foo x) (foo y)))))
                        (goo 3)))))))
```

Solución.

a) Sustitución.
(let (x 2) [x:=2]
(let (foo (lambda (a) (+ x[x:=2] 2))) [x:=2]
(let (y 3) [x:=2]
(let (foo (lambda (b) (- y b))) [x:=2]
(let (x 4) [x:=2]
(let (goo (lambda (b) (+ (foo x) (foo y))))) [x:=2]
(goo 3)))))) [x:=2]

(let (x 2)
(let (foo (lambda (a) (+ 2 2))) [foo:=(+ 2 2)]
(let (y 3) [foo:=(+ 2 2)]
(let (foo (lambda (b) (- y b))) [foo:=(+ 2 2)]

```
        (let (x 4) [foo:=(+ 2 2)]
            (let (goo (lambda (b) (+ (foo x) (foo y))))) [foo:=(+ 2 2)]
                (goo 3)))))) [foo:=(+ 2 2)]

(let (x 2)
    (let (a 4)
        (let (y 3) [y:=3]
            (let (foo (lambda (b) (- y[y:=3] b))) [y:=3]
                (let (x 4) [y:=3]
                    (let (goo (lambda (b) (+ (foo x) (foo y))))) [y:=3]
                        (goo 3)))))) [y:=3]

(let (x 2)
    (let (a 4)
        (let (y 3)
            (let (foo (lambda (b) (- 3 b))) [foo:=(- 3 b)]
                (let (x 4) [foo:=(- 3 b)]
                    (let (goo (lambda (b) (+ (foo x) (foo y))))) [foo:=(- 3 b)]
                        (goo 3)))))) [foo:=(- 3 b)]

(let (x 2)
    (let (a 4)
        (let (y 3)
            (let (b -3)
                (let (x 4) [x:=4]
                    (let (goo (lambda (b) (+ (foo x) (foo y))))) [x:=4]
                        (goo 3)))))) [x:=4]

(let (x 2)
    (let (a 4)
        (let (y 3)
            (let (b -3)
                (let (x 4)
                    (let (goo (lambda (b) (+ (foo x) (foo y))))) [goo:=(+ (foo x) (foo y))]
                        (goo 3)))))) [goo:=(+ (foo x) (foo y))]

(let (x 2)
    (let (a 4)
        (let (y 3)
            (let (b -3)
                (let (x 4)
                    (let (+ (foo x) (foo y)))))
                        ((+ (foo x) (foo y)) 3))))
```

```
(let (x 2)
    (let (a 4)
        (let (y 3)
            (let (b -3)
                (let (x 4)
                    (let (+ (foo x) (foo y)))))
                        (+ (+ 2 3) 3))))
```

```
(let (x 2)
    (let (a 4)
        (let (y 3)
            (let (b -3)
                (let (x 4)
                    (let (+ (foo x) (foo y)))))
                        (8))))
```

b) Dinámico.
Primero hacemos la pila con los valores establecidos en los *let* quedando de la siguiente manera.

x	3
goo	(lambda (b) (+ (foo x) (foo y)))
x	4
foo	(lambda (b) (- y b))
y	3
foo	(lambda (a) (+ x 2))
x	2

A continuación tomamos la primera función de la expresión.
$\rightarrow$ (goo 3)
$\rightarrow$ ((lambda (b) (+ (foo x) (foo y))) 3)
Trabajamos ahora sobre el cuerpo de la función.
$\rightarrow$ (+ (foo x) (foo y))
$\rightarrow$ (+ ((lambda (a) (+ x 2)) x) ((lambda (a) (+ x 2)) y))
$\rightarrow$ (+ ((lambda (a) (+ 3 2)) 3) ((lambda (a) (+ 3 2)) 3))
$\rightarrow$ (+ ((lambda (a) 5) 3) ((lambda (a) 5) 3))
$\rightarrow$ (+ ((lambda (a) 5) 3) ((lambda (a) 5) 3))
$\rightarrow$ (+ 3 3)
$\rightarrow$ 6
Por lo tanto, la evaluación de la expresión en ambiente dinámico es 6.

c) Estático.
Primero hacemos la pila con los valores establecidos en los *let* quedando de la siguiente manera.

x	3
goo	(lambda (b) (+ (foo x) (foo y)))
x	4
foo	(lambda (b) (- y b))
y	3
foo	(lambda (a) (+ x 2))
x	2

A continuación tomamos la primera función de la pila y hacemos la cerradura.
→ (goo 3)
→ (<x, (+ (foo x) (foo y)), (x 2)(y 3)> -10)
Trabajamos ahora sobre el cuerpo de la cerradura.
→ (+ (foo x) (foo y))
→ (+ ((lambda (b) (- y b)) x) ((lambda (b) (- y b)) y))
→ (+ ((lambda (b) (- 3 b)) 2) ((lambda (b) (- 3 b)) 3))
→ (+ 2 3)
→ 5
Por lo tanto, la evaluación de la expresión en ambiente estático es 5.

5. ¿Por qué fue necesario introducir *cerraduras* para evaluar expresiones con alcance estático en nuestras reglas semánticas?
Solución. Porque en la cerradura almacenaremos el parámetro, cuerpo y ambiente en donde la expresión fue definida, de tal forma que al evaluar la función, en vez de que simplemente se regrese, se regresa una cerradura la cual encierra a la función con el ambiente actual.