Tarea 1:
Lenguajes de Programación

Araujo Chavez Mauricio
312210047

Carmona Mendoza Matín
313075977

1. Los naturales de Church se definen como sigue:

$$\begin{aligned} 0 &= \lambda s.\lambda z.z \\ 1 &= \lambda s.\lambda z.s\ z \\ 2 &= \lambda s.\lambda z.s(s\ z) \\ 3 &= \lambda s.\lambda z.s(s(s\ z)) \\ &\vdots \end{aligned}$$

Se define el par ordenado como pair:= $\lambda x.\lambda y.\lambda p.p\ x\ y$, así el par ordenado (a,b)= $\underline{pair}\ \ a\ b = \lambda p.p\ a\ b$ Las funciones para obtener la primer y segunda componente de un par ordenado se definen respetivamente como: fst:= $\lambda p.p\ \underline{true}$ y snd:= $\lambda p.p\ \underline{false}$
Sean g_1, h_1 las siguientes funciones:

$$g_1 := \lambda n.\lambda s.\lambda z.n\ (\lambda h_1.\lambda h_2.\ h2\ (h1\ s))\ (\lambda u.z)\ (\lambda u.u)$$

$$h_1 := \lambda n.\underline{fst}\ (n\ \underline{ss}\ \underline{zz}), \text{ donde } \underline{\text{ss}} = \lambda p.\underline{pair}\ (\underline{snd}\ p)\ (\underline{suc}\ (\underline{snd}\ p)), \text{ y } \underline{zz} := \underline{pair}\ 0\ 0$$

a) Calcula $(g_1\ 0)$ y $(g_1\ 3)$

- $(g_1\ 0)$
  $\lambda n.\lambda s.\lambda z.n(\lambda h1.\lambda h2.h2\ (h1\ s))(\lambda u.z)(\lambda u.u)\ 0 =$
  $\lambda s.\lambda z.0(\lambda h1.\lambda h2.h2\ (h1\ s))(\lambda u.z)(\lambda u.u)\ =$
  $\lambda s.\lambda z.(\lambda s.\lambda z.z)(\lambda h1.\lambda h2.h2\ (h1\ s))(\lambda u.z)(\lambda u.u)\ =$
  $\lambda s.\lambda z.(\lambda z.z)(\lambda u.z)(\lambda u.u)\ =$
  $\lambda s.\lambda z.(\lambda u.z)(\lambda u.u)\ =$
  $\lambda s.\lambda z.z\ = 0$

- $(g_1\ 3)$
  $\lambda n.\lambda s.\lambda z.n(\lambda h1.\lambda h2.h2\ (h1\ s))(\lambda u.z)(\lambda u.u)\ 3 =$
  $\lambda s.\lambda z.3(\lambda h1.\lambda h2.h2\ (h1\ s))(\lambda u.z)(\lambda u.u) =$
  $\lambda s.\lambda z.(\lambda s.\lambda z.s(s(s\ z)))(\lambda h1.\lambda h2.h2\ (h1\ s))(\lambda u.z)(\lambda u.u) =$
  $\lambda s.\lambda z.(\lambda s^2.\lambda z^2.s^2(s^2(s^2\ z^2)))(\lambda h1.\lambda h2.h2\ (h1\ s))(\lambda u.z)(\lambda u^2.u^2) =$
  $\lambda s.\lambda z.(\lambda z^2.((\lambda h1.\lambda h2.h2\ (h1\ s)))(((\lambda h1.\lambda h2.h2\ (h1\ s)))(((\lambda h1.\lambda h2.h2\ (h1\ s)))\ z^2)))(\lambda u.z)(\lambda u^2.u^2) =$
  $\lambda s.\lambda z.(\lambda z^2.((\lambda h1.\lambda h2.h2\ (h1\ s)))(((\lambda h1^2.\lambda h2^2.h2^2\ (h1^2\ s)))$
  $(((\lambda h1^3.\lambda h2^3.h2^3\ (h1^3\ s)))\ z^2)))(\lambda u.z)(\lambda u^2.u^2) =$
  $\lambda s.\lambda z.((\lambda h1.\lambda h2.h2\ (h1\ s)))(((\lambda h1^2.\lambda h2^2.h2^2\ (h1^2\ s)))(((\lambda h1^3.\lambda h2^3.h2^3\ (h1^3\ s)))\ (\lambda u.z)))(\lambda u^2.u^2) =$
  $\lambda s.\lambda z.((\lambda h2.h2\ ((((\lambda h1^2.\lambda h2^2.h2^2\ (h1^2\ s)))(((\lambda h1^3.\lambda h2^3.h2^3\ (h1^3\ s)))\ (\lambda u.z)))s)))(\lambda u^2.u^2) =$
  $\lambda s.\lambda z.(((\lambda u^2.u^2)\ ((((\lambda h1^2.\lambda h2^2.h2^2\ (h1^2\ s)))(((\lambda h1^3.\lambda h2^3.h2^3\ (h1^3\ s)))\ (\lambda u.z)))s))) =$
  $\lambda s.\lambda z.((\lambda h1^2.\lambda h2^2.h2^2\ (h1^2\ s)))(((\lambda h1^3.\lambda h2^3.h2^3\ (h1^3\ s)))\ (\lambda u.z)))s))) =$
  $\lambda s.\lambda z.((\lambda h2^2.h2^2\ ((((\lambda h1^3.\lambda h2^3.h2^3\ (h1^3\ s)))\ (\lambda u.z)))s)))\ s))) =$
  $\lambda s.\lambda z.((s\ ((((\lambda h1^3.\lambda h2^3.h2^3\ (h1^3\ s)))\ (\lambda u.z)))s)) =$
  $\lambda s.\lambda z.(s((((\lambda h2^3.h2^3\ ((\lambda u.z)\ s)))s)) =$
  $\lambda s.\lambda z.(s(s((\lambda u.z)\ s))) =$
  $\lambda s.\lambda z.(s(s\ z)) = 2$

b) Calcula $(h_1\ 1)$ y $(h_1\ 2)$

- $(h_1\ 1)$
  $(\lambda n.\underline{fst}(n\ \underline{ss}\ \underline{zz}))\ 1 =$
  $(\underline{fst}(1\ \underline{ss}\ \underline{zz})) =$
  $(\underline{fst}((\lambda s.\lambda z.s\ z)\ \underline{ss}\ \underline{zz})) =$
  $(\underline{fst}((\lambda z.\underline{ss}\ z)\ \underline{zz})) =$
  $\underline{fst}(\underline{ss}\ \underline{zz}) =$
  $\underline{fst}(\underline{ss}\ (\underline{pair}\ 0\ 0)) =$
  $\underline{fst}(\underline{ss}\ (\lambda p.p\ 0\ 0)) =$
  $\underline{fst}(\lambda p.\underline{pair}(\underline{snd}\ p)(\underline{suc}(\underline{snd}\ p))\ (\lambda p.p\ 0\ 0)) =$

$\underline{fst}(\underline{pair}(\underline{snd}\ (\lambda p.p\ 0\ 0))(\underline{suc}(\underline{snd}\ (\lambda p.p\ 0\ 0)))) =$
$\underline{fst}(\underline{pair}\ 0\ (\underline{suc}(\underline{snd}\ (\lambda p.p\ 0\ 0)))) =$
$\underline{fst}(\underline{pair}\ 0\ (\underline{suc}\ 0)) =$
$\underline{fst}(\underline{pair}\ 0\ 1) =$
$\underline{fst}(\lambda p.p\ 0\ 1) = 0$

- $(h_1\ 2)$
  $(\lambda n.\underline{fst}(n\ \underline{ss}\ \underline{zz}))\ 2 =$
  $\underline{fst}(2\ \underline{ss}\ \underline{zz})) =$
  $\underline{fst}(((\lambda s.\lambda z.s(s\ z))\ \underline{ss}\ \underline{zz}) =$
  $\underline{fst}((\lambda z.\underline{ss}(\underline{ss}\ z))\ \underline{zz})) =$
  $\underline{fst}(\underline{ss}(\underline{ss}\ \underline{zz})) =$
  $\underline{fst}(\underline{ss}(\underline{pair}\ 0\ 1)) =$
  $\underline{fst}((\lambda p.\underline{pair}(\underline{snd}\ p)(\underline{suc}(\underline{snd}\ p))(\underline{pair}\ 0\ 1)) =$
  $\underline{fst}(\underline{pair}(\underline{snd}\ (\underline{pair}\ 0\ 1))(\underline{suc}(\underline{snd}\ (\underline{pair}\ 0\ 1)))) =$
  $\underline{fst}(\underline{pair}(\underline{snd}\ (\underline{pair}\ 0\ 1))(\underline{suc}\ 1)) =$
  $\underline{fst}(\underline{pair}(\underline{snd}\ (\underline{pair}\ 0\ 1))\ 2) =$
  $\underline{fst}(\underline{pair}\ 1\ 2) = 1$

c) ¿Qué hacen las funciones g_1 y h_1?

Las funciones g_1 y h_1 calculan el antecesor de un número (En caso de que el número sea 0 la función regresa 0).

2. Los naturales de Scott se definen como sigue:

$$0 = \lambda x.\lambda y.x$$
$$1 = \lambda x.\lambda y.y\ 0$$
$$2 = \lambda x.\lambda y.y\ 1$$
$$3 = \lambda x.\lambda y.y\ 2$$
$$\vdots$$

Sean f_2, g_2 y h_2 las siguientes funciones:

$$f_2 := \lambda n.\lambda x.\lambda y.y\ n$$
$$g_2 := \lambda n.n\ 0\ (\lambda x.x)$$
$$h_2 := \lambda n.n\ \underline{true}\ (\lambda x.\underline{false})$$

a) Calcula $(f_2\ 0)$ y $(f_2\ 3)$

- $(f_2\ 0)$
  $(\lambda n.\lambda x.\lambda y.y\ n)\ 0 =$
  $\lambda x.\lambda y.y\ 0 =$
  $\lambda x.\lambda y.y\ (\lambda x.\lambda y.x) = 1$

- $(f_2\ 3)$
  $(\lambda n.\lambda x.\lambda y.y\ n)\ 3 =$
  $\lambda x.\lambda y.y(\lambda x.\lambda y.y\ 2) =$
  $\lambda x.\lambda y.y(\lambda x.\lambda y.y\ (\lambda x.\lambda y.y\ 1)) =$
  $\lambda x.\lambda y.y(\lambda x.\lambda y.y\ (\lambda x.\lambda y.y\ (\lambda x.\lambda y.y\ 0))) =$
  $\lambda x.\lambda y.y(\lambda x.\lambda y.y\ (\lambda x.\lambda y.y\ (\lambda x.\lambda y.y\ (\lambda x.\lambda y.x)))) = 4$

b) Calcula $(g_2\ 1)$ y $(g_2\ 4)$

- $(g_2\ 1)$
  $(\lambda n.n\ 0\ (\lambda x.x))\ 1$
  $(1\ 0\ (\lambda x.x)) =$
  $((\lambda x.\lambda y.y\ (\lambda x.\lambda y.x))\ 0\ (\lambda x.x)) =$
  $((\lambda y.y\ (\lambda x.\lambda y.x))\ (\lambda x.x)) =$
  $(\lambda x.x)\ (\lambda x.\lambda y.x) =$
  $(\lambda x.\lambda y.x) = 0$
- $(g_2\ 4)$
  $(\lambda n.n\ 0\ (\lambda x.x))\ 4$
  $(4\ 0\ (\lambda x.x))\ 1$
  $((\lambda x.\lambda y.y\ (\lambda x.\lambda y.y\ (\lambda x.\lambda y.y(\lambda x.\lambda y.y(\lambda x.\lambda y.x)))))\ 0\ (\lambda x.x)) =$
  $((\lambda y.y\ (\lambda x.\lambda y.y\ (\lambda x.\lambda y.y(\lambda x.\lambda y.y(\lambda x.\lambda y.x)))))\ (\lambda x.x)) =$
  $(\lambda x.x)\ (\lambda x.\lambda y.y\ (\lambda x.\lambda y.y(\lambda x.\lambda y.y(\lambda x.\lambda y.x)))) =$
  $(\lambda x.\lambda y.y\ (\lambda x.\lambda y.y(\lambda x.\lambda y.y(\lambda x.\lambda y.x)))) = 3$

c) Calcula $(h_2\ 0)$ y $(h_2\ 5)$

- $(h_2\ 0)$
  $(\lambda n.n\ \underline{true}\ (\lambda x.\underline{false}))\ 0 =$
  $(0\ \underline{true}\ (\lambda x.\underline{false})) =$
  $((\lambda x.\lambda y.x)\ \underline{true}\ (\lambda x.\underline{false})) =$
  $((\lambda y.\underline{true})\ (\lambda x.\underline{false})) = \underline{true}$

- $(h_2\ 5)$
  $(\lambda n.n\ \underline{true}\ (\lambda x.\underline{false}))\ 5 =$
  $(5\ \underline{true}\ (\lambda x.\underline{false})) =$
  $((\lambda x.\lambda y.y(\lambda x.\lambda y.y(\lambda x.\lambda y.y(\lambda x.\lambda y.y(\lambda x.\lambda y.y(\lambda x.\lambda y.x))))))\ \underline{true}\ (\lambda x.\underline{false})) =$
  $((\lambda y.y(\lambda x.\lambda y.y(\lambda x.\lambda y.y(\lambda x.\lambda y.y(\lambda x.\lambda y.y(\lambda x.\lambda y.x))))))\ (\lambda x.\underline{false})) =$
  $(((\lambda x.\underline{false})(\lambda x.\lambda y.y(\lambda x.\lambda y.y(\lambda x.\lambda y.y(\lambda x.\lambda y.y(\lambda x.\lambda y.x))))))) = \underline{false}$

d) ¿Qué hacen las funciones f_2, g_2 y h_2?

- La función f_2 nos da el sucesor del número n
- La función g_2 nos da el predecesor del número n
- La función h_2 regresa <u>true</u> si n es cero y <u>false</u> si n no es cero.

e) (Extra [+1 punto]) Haz una función que haga la suma de naturales de Scott

$V\ (Combinador\ Turing) : (\lambda x.\lambda y.(y((xx)y)))\ (\lambda x.\lambda y.(y((xx)y)))$
$suc\ (sucesor) : \lambda n.\lambda x.\lambda y.y\ n$
$sumaScott : V\ (\lambda f.\lambda m.\lambda n.(m\ n)\ (\lambda m'.\underline{suc}\ ((f\ m')\ n)))$

4. Utilizando un combinador de punto fijo, implementa estas funciones de forma recursiva:

a) **Una función que dados n y m calcule n^m.**

Sobre los naturales de Church

Para esto utilizamos las siguientes funciones, definidas de la siguiente manera:;

$ift\ (if - then - else) : \lambda b.\lambda t.\lambda e.(b\ t)\ e$

$true : \lambda x.\lambda y.x$

$false : \lambda x.\lambda y.y$

$esCero? : \lambda n.(n\ (\lambda x.\underline{\text{false}}))\ \underline{\text{true}}$

$suc\ (sucesor) : \lambda n.\lambda s.\lambda z.s\ ((n\ s)\ z)$

$suma : \lambda n.\lambda m.(n\ \underline{\text{suc}})\ m$

$prod\ (producto) : \lambda n.\lambda m.(n\ (\underline{\text{suma}}\ m))\ (\lambda s.\lambda z.z)$

$Y\ (Combinador\ Y) : \lambda f.(\lambda x.f(x\ x))\ (\lambda x.f(x\ x))$

$pred\ (predecesor) : g_1$ (1er Ejercicio)

$expAux : \lambda f.\lambda n.\lambda m.\underline{\text{ift}}\ (\underline{\text{esCero?}}\ m)\ (\lambda s.\lambda z.s\ z)\ (\underline{\text{prod}}\ n\ (f\ n\ (\underline{\text{pred}}\ m)))$

FuncionExponente : $Y\ \underline{\text{expAux}}$

b) **Una función que decida si un natural de Church es impar.**

$caseN : \lambda n.\lambda a.\lambda f.(n\ a)\ f$
(Si 'n' es cero de Church, regresa 'a', si no regresa (f x), donde x es el predecesor de n.)

$impSAux : \lambda f.\lambda n.\underline{\text{caseN}}\ n\ \underline{\text{false}}\ (\lambda n'.\underline{\text{caseN}}\ n'\ \underline{\text{true}}\ f)$

FuncionImparS : $Y\ \underline{\text{impSAux}}$

5. Da una definicion inductiva mediante juicios de palabras palíndromas sobre el alfabeto a, b.

a) Enuncia el principio de induccion para los juicios que definiste.

$\overline{aP}\ \ \overline{bP}$

$\dfrac{wP}{awaP}\ \ \dfrac{wP}{bwbP}$

Las primeras dos son un axioma; la cadena de un sólo elemento es palindroma.
Las siguientes dos nos dice que si w es una cadena palindroma entonces al agregar una a al principio y otra al final (o una b) sigue siendo palindroma.

b) Demuestra por inducción matemática que si w es una cadena palíndroma entonces reverse(w) = w.

7. Extiende el lenguaje EAB con un operador even que tome un natural y decida si dicho numero es par de la siguiente manera.

■Extiende la sintaxis concreta

e ::= x | n | true | false | e+e | e*e | suc e | pred e | if e then e else e|
iszero e | iseven e | let x = e in e end

■Extiende la sintaxis abstracta

t ::= x | num[n] | bool[true] | bool[false] | suma(t1,t2) | prod(t1,t2) | suc(t) | pred(t) | if(t1,t2,t3) | iszero(t) | iseven(t) | let(t1,x.t2)

■Extiende la semántica estática

Γ ⊢ t : Nat
Γ ⊢ iseven t: Bool

■Extiende la semántica dinámica

$$\frac{}{isevennum[0] \to bool[true]}$$

$$\frac{}{isevennum[1] \to bool[false]}$$

$$\frac{t1 \to t1'}{iseven t1 \to iseven t1'}$$

$$\frac{}{isevennum[n] \to iseven[pred(pred(num[n]))]}$$

8. Sean e1 y e2 las siguientes expresiones:

e1 = let y = suc(2+1) in (let z = pred(5) in z+y end) * 2+y end
e2 = (let y = x+v in (let z = x in x*y*z end) end)[x:=y*z]

- Convierte las expresiones e1 y e2 a sus respectivas representaciones como asas, t1 y t2, respectivamente.
  t1:=let(var[y],suc(suma(num[2],num[1])),prod(let(var[z],pred(num[5]), suma(var[z],var[y])),suma(num[2],var[y])))
  t2:=let z = let(var[z],var[x],prod(var[x],prod(var[y],var[z])))
  t2':=let y = let(var[y],suma(var[x],var[v]),let(var[z],var[x],prod(var[x],prod(var[y],var[z])))) [x:=y*z]
- Con los juicios para la semantica estática haz derivaciones para t1 y t2.
  e1 =let y = suc(2+1) in (let z = pred(5) in z+y end) * 2+y end
  t1=let(var[y],suc(suma(num[2],num[1])),
  prod(let(var[z],pred(num[5]),suma(var[z],var[y])),suma(num[2],var[y])))

  ⊢ num[2]:Nat (tnum)
  ⊢ num[1]:Nat (tnum)
  ⊢ suma(num[2],num[1]): Nat (tsum)
  ⊢ suc(suma(num[2],num[1])):Nat (tsuc)
  y:Nat ⊢ y:Nat (tvar)
  y:Nat,z:Nat ⊢ z:Nat (tvar)
  y:Nat,z:Nat ⊢ num[5]:Nat (tnum)
  y:Nat,z:Nat ⊢ pred(num[5]):Nat (tpred)
  y:Nat,z:Nat ⊢ suma(var[z],var[y]): Nat (tsum)
  y:Nat,z:Nat ⊢ let(var[z],pred(num[5]),suma(var[z],var[y])):Nat (tlet)
  y:Nat,z:Nat ⊢ suma(num[2],var[y]):Nat (tsum)
  y:Nat,z:Nat ⊢ prod(suma(var[z],var[y]),suma(num[2],var[y])):Nat (tprod)
  y:Nat,z:Nat ⊢let(var[y], suc(suma(num[2],num[1])),

prod(let(var[z],pred(num[5]),
suma(var[z],var[y])),suma(num[2],var[y])))):Nat (tlet)

e2 = (let y = x+v in (let z = x in x*y *z end) end)[x:=y *z]
t2 =let y = let(var[y],suma(var[x],var[v]),let(var[z],var[x],prod(var[x],prod(var[y],var[z])))) [x:=y*z]

x:T ⊢ x:T
v:S ⊢ v:S
x:T,v:S ⊢ suma(var[x],var[v]):Nat

- Evalua t1 y t2 usando los juicios para la semantica dinámica.

  t1=let(var[y],suc(suma(num[2],num[1])),prod(let(var[z],pred(num[5]),
  suma(var[z],var[y])),suma(num[2],var[y])))
  e1= let y = suc(2+1) in (let z = pred(5) in z+y end) * 2+y end

  let y = suc(2+1) in (let z = pred(5) in z+y end) * 2+y end (eleti)
  let y = suc(3) in (let z = pred(5) in z+y end) * 2+y end (eleti)
  let y = 4 in (let z = pred(5) in z+y end) * 2+y end (eletf)
  ((let z = pred(5) in z+y end)*2+y)[y:=4] =
  (let z = pred(5) in z+4 end) * (2 + 4) (eleti)
  (let z = 4 in z+4) * (2+4) (eletf)
  (z+4)[z:=4]*(2+4) =
  (4+4)*(2+4) (eprodi)(esumaf)
  16*(2+4) (eprodd)(esumaf)
  16*6 (eprodf)
  96

  t2 =let y = let(var[y],suma(var[x],var[v]),let(var[z],var[x],prod(var[x],prod(var[y],var[z])))) [x:=y*z]
  e2 = (let y = x+v in (let z = x in x*y *z end) end)[x:=y *z]

  let y = y*z+v in (let z = y*z in y*z*y*z) (eletf)
  (let z = y*z in y*z*y*z) [y:=z*z+v] =
  let z = (z*z+v)*z in (z*z+v)*z*(z*z+v)*z (eletf)
  ((z*z+v)*z*(z*z+v)*z*)[z:=z*z+v]