Ejercicio 1

a) b)

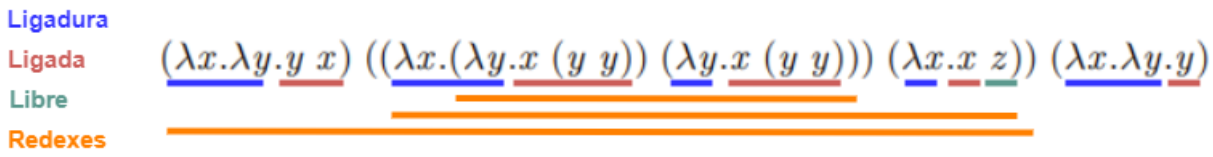

Figure 1: Ligaduras, ocurrencias ligadas, libres y redexes.

c)

Por el corolario del teorema de Church-Rosser si el termino tiene forma normal entonces es única, como este caso tiene forma normal (ver d)), entonces tiene una única forma normal.

d)

Se aplicará call by name.

$$
\begin{aligned}
t &= (\lambda x.\lambda y.y\ x)\ ((\lambda x.(\lambda y.x\ (y\ y))\ (\lambda y.x\ (y\ y)))\ (\lambda x.x\ z))\ (\lambda x.\lambda y.y) \\
&\rightarrow_\beta (\lambda y.y\ ((\lambda x.(\lambda y.x\ (y\ y))\ (\lambda y.x\ (y\ y)))\ (\lambda x.x\ z)))\ (\lambda x.\lambda y.y) \\
&\rightarrow_\beta (\lambda x.\lambda y.y)\ ((\lambda x.(\lambda y.x\ (y\ y))\ (\lambda y.x\ (y\ y)))\ (\lambda x.x\ z))) \\
&\rightarrow_\beta (\lambda y.y)
\end{aligned}
$$

Ejercicio 2

$Pair\ a\ b = \forall \alpha.\ (a \to b \to \alpha) \to \alpha$
$Nat = \forall \alpha.\ (\alpha \to \alpha) \to \alpha \to \alpha$

a)

$SWAP\ :\ Pair\ a\ b \to Pair\ b\ a$
$SWAP = \lambda p^{Pair\ a\ b}.\ PAIR\ (SND\ p)\ (FST\ p)$

b)

$DUP\ :\ Nat \to\ Nat$
$DUP = \lambda n^{Nat}.\ foldN\ Nat\ S\ n\ n$

c)

$EXP2\ :\ Nat \to\ Nat$
$EXP2 = \lambda n^{Nat}.\ foldN\ Nat\ (PLUS\ n)\ Z\ n$

Ejercicio 3

La solución se encuentra en el archivo `ej3.hs`.

Ejercicio 4

Antes de mostrar los resultados de ejecutar el comando `ghc -ddump-simpl` para cada archivo, se explicarán algunos conceptos. El pragma *{-# NOINLINE g #-}* sirve para que el compilador no sea "inlined", por lo que se pudo observar quitando y agregando el pragma en distintos casos, al no utilizarlo la función h aparecía como que no utilizaba a la función g. Suponemos que esto se daba porque realizaba varias simplificaciones y decidía que no era necesario agregarla.

Por otro lado, también se utiliza *{-# LANGUAGE RankNTypes #-}*, en este caso se utiliza para poder utilizar `forall`.

A continuación se mostrarán los resultados de ejecutar el comando indicado en cada sección.

a) `ghc -ddump-simpl Pru1.hs`

```
[1 of 1] Compiling Pru1             ( Pru1.hs, Pru1.o )

==================== Tidy Core ====================
Result size of Tidy Core
= {terms: 25, types: 19, coercions: 0, joins: 0/0}

-- RHS size: {terms: 5, types: 5, coercions: 0, joins: 0/0}
g [InlPrag=NOINLINE] :: forall a. (a -> a) -> a -> a
[GblId, Arity=2, Caf=NoCafRefs, Unf=OtherCon []]
g = \ (@ a_atL) (f_agd :: a_atL -> a_atL) (a1_age :: a_atL) ->
    f_agd a1_age

-- RHS size: {terms: 4, types: 2, coercions: 0, joins: 0/0}
h :: Int
[GblId]
h = g @ Int (id @ Int) (GHC.Types.I# 4#)

...
```

Se encontraron las siguientes similitudes con System F:

Output:

```
g [InlPrag=NOINLINE] :: forall a. (a -> a) -> a -> a
g = \ (@ a_atL) (f_agd :: a_atL -> a_atL) (a1_age :: a_atL) ->
    f_agd a1_age
```

System F:

$g : \forall a.(a -> a) -> a -> a$
$g = \Lambda a\_atL.\lambda f\_agd^{a\_atL->a\_atL}.\lambda a1\_age^{a\_atL}.f\_agd\ a1\_age$
Output:

```
h :: Int
h = g @ Int (id @ Int) (GHC.Types.I# 4#)
```

System F:
$h : Int$
$h = g\ Int\ (id\ Int)\ 4^{Int}$
Luego el código que resta se encargaba de aspectos de los módulos de haskell y tipado.

b) `ghc -ddump-simpl Pru2.hs`

```
[1 of 1] Compiling Pru2             ( Pru2.hs, Pru2.o )

==================== Tidy Core ====================
Result size of Tidy Core
= {terms: 67, types: 30, coercions: 0, joins: 0/0}

-- RHS size: {terms: 5, types: 5, coercions: 0, joins: 0/0}
g [InlPrag=NOINLINE] :: forall a. (a -> a) -> a -> a
[GblId, Arity=2, Caf=NoCafRefs, Unf=OtherCon []]
g = \ (@ a_agO) (f_agg :: a_agO -> a_agO) (a1_agh :: a_agO) ->
    f_agg a1_agh

-- RHS size: {terms: 3, types: 2, coercions: 0, joins: 0/0}
h :: Foo
[GblId]
h = g @ Foo (id @ Foo) Pru2.C1

...
```

Se encontraron las siguientes similitudes con System F:
Output:

```
g [InlPrag=NOINLINE] :: forall a. (a -> a) -> a -> a
g = \ (@ a_agO) (f_agg :: a_agO -> a_agO) (a1_agh :: a_agO) ->
    f_agg a1_agh
```

System F:

$g : \forall a.(a -> a) -> a -> a$

$g = \Lambda a\_atL.\lambda f\_agd^{a\_atL \text{ -> } a\_atL}.\lambda a1\_age^{a\_atL}.f\_agd\ a1\_age$

Output:

```
h :: Foo
h = g @ Foo (id @ Foo) Pru2.C1
```

System F:

$h : Foo$

$h = g\ Foo\ (id\ Foo)\ C1^{Foo}$

Luego el código que resta se encargaba de aspectos de los módulos de haskell, tipado y esta vez también la creación del data Foo y sus constructores.

c) `ghc -ddump-simpl Pru3.hs`

```
[1 of 1] Compiling Pru3             ( Pru3.hs, Pru3.o )

==================== Tidy Core ====================
Result size of Tidy Core
= {terms: 32, types: 37, coercions: 0, joins: 0/0}

-- RHS size: {terms: 10, types: 14, coercions: 0, joins: 0/0}
g [InlPrag=NOINLINE]
:: forall a b. (forall a1. a1 -> a1) -> a -> b -> (a, b)
[GblId, Arity=3, Caf=NoCafRefs, Unf=OtherCon []]
g = \ (@ a_aA8)
    (@ b_aA9)
    (f_agf :: forall a1. a1 -> a1)
    (a1_agg :: a_aA8)
    (b1_agh :: b_aA9) ->
    (f_agf @ a_aA8 a1_agg, f_agf @ b_aA9 b1_agh)

-- RHS size: {terms: 6, types: 2, coercions: 0, joins: 0/0}
h :: (Int, Char)
[GblId]
h = g @ Int @ Char id (GHC.Types.I# 4#) (GHC.Types.C# 'a'#)

...
```

Se encontraron las siguientes similitudes con System F:

Output:

```
g [InlPrag=NOINLINE]
:: forall a b. (forall a1. a1 -> a1) -> a -> b -> (a, b)
g = \ (@ a_aA8)
    (@ b_aA9)
    (f_agf :: forall a1. a1 -> a1)
    (a1_agg :: a_aA8)
    (b1_agh :: b_aA9) ->
    (f_agf @ a_aA8 a1_agg, f_agf @ b_aA9 b1_agh)
```

System F:

$g : \forall a.\forall b.(\forall a1.a1 -> a1) -> a -> b -> (a, b)$

$g = \Lambda a\_aA8.\Lambda b\_aA9.(\forall a1.\lambda f\_agf^{a1->a1}).\lambda a1\_agg^{a\_aA8}.\lambda b1\_agh^{b\_aA9}.(f\_agf\ a\_aA8\ a1\_agg, f\_agf\ b\_aA9\ b1\_agh)$

Output:

```
h :: (Int, Char)
h = g @ Int @ Char id (GHC.Types.I# 4#) (GHC.Types.C# 'a'#)
```

System F:

$h : (Int, Char)$

$h = g\ Int\ Char\ id\ 4^{Int}\ a^{Char}$

Así como en los casos anteriores, el resto del output refería a la creación del módulo y tipado de haskell.

d) `ghc -ddump-simpl Pru4.hs`

```
[1 of 1] Compiling Pru4             ( Pru4.hs, Pru4.o )

==================== Tidy Core ====================
Result size of Tidy Core
= {terms: 110, types: 86, coercions: 32, joins: 0/0}

-- RHS size: {terms: 3, types: 4, coercions: 2, joins: 0/0}
g [InlPrag=INLINE] :: forall a. G a => a -> Int
[GblId[ClassOp],
Arity=1,
```

```
    Caf=NoCafRefs,
    Str=<S,U>,
    Unf=Unf{Src=InlineStable, TopLvl=True, Value=True, ConLike=True,
            WorkFree=True, Expandable=True,
            Guidance=ALWAYS_IF(arity=1,unsat_ok=False,boring_ok=True)
            Tmpl= \ (@ a_agd) (v_B1 [Occ=Once1] :: G a_agd) ->
                    v_B1 `cast` (Pru4.N:G[0] <a_agd>_N :: G a_agd ~R# (a_agd -> Int))}]
g = \ (@ a_agd) (v_B1 :: G a_agd) ->
    v_B1 `cast` (Pru4.N:G[0] <a_agd>_N :: G a_agd ~R# (a_agd -> Int))

-- RHS size: {terms: 6, types: 2, coercions: 0, joins: 0/0}
$cg_rHC :: Int -> Int
[GblId, Arity=1, Unf=OtherCon []]
$cg_rHC
= \ (x_auN :: Int) ->
    + @ Int GHC.Num.$fNumInt x_auN (GHC.Types.I# 1#)

-- RHS size: {terms: 1, types: 0, coercions: 3, joins: 0/0}
Pru4.$fGInt [InlPrag=INLINE (sat-args=0)] :: G Int
[GblId[DFunId(nt)], Arity=1, Unf=OtherCon []]
Pru4.$fGInt
= $cg_rHC
    `cast` (Sym (Pru4.N:G[0] <Int>_N) :: (Int -> Int) ~R# G Int)

-- RHS size: {terms: 14, types: 17, coercions: 4, joins: 0/0}
$cg1_rIa :: forall a b. (G a, G b) => (a, b) -> Int
[GblId, Arity=3, Unf=OtherCon []]
$cg1_rIa
= \ (@ a_aBW)
    (@ b_aBX)
    ($dG_aBY :: G a_aBW)
    ($dG1_aBZ :: G b_aBX)
    (ds_dHQ :: (a_aBW, b_aBX)) ->
    case ds_dHQ of { (x_atz, y_atA) ->
    + @ Int
        GHC.Num.$fNumInt
        (($dG_aBY
        `cast` (Pru4.N:G[0] <a_aBW>_N :: G a_aBW ~R# (a_aBW -> Int)))
        x_atz)
        (($dG1_aBZ
        `cast` (Pru4.N:G[0] <b_aBX>_N :: G b_aBX ~R# (b_aBX -> Int)))
        y_atA)
```

```
    }

-- RHS size: {terms: 1, types: 0, coercions: 17, joins: 0/0}
Pru4.$fG(,) [InlPrag=INLINE (sat-args=0)]
:: forall a b. (G a, G b) => G (a, b)
[GblId[DFunId(nt)], Arity=3, Unf=OtherCon []]
Pru4.$fG(,)
= $cg1_rIa
    `cast` (forall (a :: <*>_N) (b :: <*>_N).
            <G a>_R ->_R <G b>_R ->_R Sym (Pru4.N:G[0] <(a, b)>_N)
            :: (forall a b. G a -> G b -> (a, b) -> Int)
            ~R# (forall a b. G a -> G b -> G (a, b)))

-- RHS size: {terms: 8, types: 4, coercions: 6, joins: 0/0}
h :: Int
[GblId]
h = $cg1_rIa
    @ Int
    @ Int
    ($cg_rHC
    `cast` (Sym (Pru4.N:G[0] <Int>_N) :: (Int -> Int) ~R# G Int))
    ($cg_rHC
    `cast` (Sym (Pru4.N:G[0] <Int>_N) :: (Int -> Int) ~R# G Int))
    (GHC.Types.I# 4#, GHC.Types.I# 5#)

...
```

Se encontraron las siguientes similitudes con System F:

Output:

```
g [InlPrag=INLINE] :: forall a. G a => a -> Int
[GblId[ClassOp],
Arity=1,
Caf=NoCafRefs,
Str=<S,U>,
Unf=Unf{Src=InlineStable, TopLvl=True, Value=True, ConLike=True,
        WorkFree=True, Expandable=True,
        Guidance=ALWAYS_IF(arity=1,unsat_ok=False,boring_ok=True)
        Tmpl= \ (@ a_agd) (v_B1 [Occ=Once1] :: G a_agd) ->
                v_B1 `cast` (Pru4.N:G[0] <a_agd>_N :: G a_agd ~R# (a_agd -> Int))}]
g = \ (@ a_agd) (v_B1 :: G a_agd) ->
    v_B1 `cast` (Pru4.N:G[0] <a_agd>_N :: G a_agd ~R# (a_agd -> Int))
```

System F:

Define la firma de la función g de la clase G $g : \forall a.Ga \vDash a -> Int$

Output:

```
$cg_rHC :: Int -> Int
[GblId, Arity=1, Unf=OtherCon []]
$cg_rHC
= \ (x_auN :: Int) ->
    + @ Int GHC.Num.$fNumInt x_auN (GHC.Types.I# 1#)
```

System F:

$cg\_rHC : Int -> Int\ cg\_rHC = x\_auN^{Int}.+^{Int}\ x\_auN1^{Int}$

Output:

```
$cg1_rIa :: forall a b. (G a, G b) => (a, b) -> Int
[GblId, Arity=3, Unf=OtherCon []]
$cg1_rIa
= \ (@ a_aBW)
    (@ b_aBX)
    ($dG_aBY :: G a_aBW)
    ($dG1_aBZ :: G b_aBX)
    (ds_dHQ :: (a_aBW, b_aBX)) ->
    case ds_dHQ of { (x_atz, y_atA) ->
    + @ Int
        GHC.Num.$fNumInt
        (($dG_aBY
        `cast` (Pru4.N:G[0] <a_aBW>_N :: G a_aBW ~R# (a_aBW -> Int)))
        x_atz)
        (($dG1_aBZ
        `cast` (Pru4.N:G[0] <b_aBX>_N :: G b_aBX ~R# (b_aBX -> Int)))
        y_atA)
    }
```

System F:

$cg1\_rIa : \forall ab.(Ga, Gb) \vDash (a, b) -> Int$

$cg1\_rIa = \Lambda a\_abW.\Lambda b\_aBX.\lambda \$dG\_aBY.\lambda \$dG1\_aBZ.\lambda ds\_dHQ^{(a\_aBW, b\_aBX)}.+^{Int} x\_atz^{Int} y\_atA^{Int}$

Output:

```
h :: Int
[GblId]
```

```
h = $cg1_rIa
    @ Int
    @ Int
    ($cg_rHC
    `cast` (Sym (Pru4.N:G[0] <Int>_N) :: (Int -> Int) ~R# G Int))
    ($cg_rHC
    `cast` (Sym (Pru4.N:G[0] <Int>_N) :: (Int -> Int) ~R# G Int))
    (GHC.Types.I# 4#, GHC.Types.I# 5#)
```

System F:

$h : Int$

$h = \$cg1\_rIa\ Int\ Int\ \$cg\_rHC\ \$cg\_rHC\ (4^{Int}, 5^{Int})$

Así como en los casos anteriores, el resto del output refería a la creación del módulo, tipado de haskell y creación de la clase y las instancias.

De las similitudes encontradas se pudo hacer un mapeo de algunos símbolos:

- @ a $\rightarrow \Lambda$ a
- (f :: a $\rightarrow$ b) $\rightarrow f^{a \rightarrow b}$
- (a :: b) $\rightarrow a^b$
- (\) $\rightarrow \lambda$ Indicaba que se recibían los parámetros de la función. Una vez que ya se tenían todos los parámetros entonces escribe ->

En el caso de (@ a) no solo se usa para definir una instancia de tipo, también se usa para indicar el tipo a utilizar de una función en el caso de ser una función polimórfica. Ejemplo:

Instanciar tipo:

```
g = \ (@ a_atL) (f_agd :: a_atL -> a_atL) (a1_age :: a_atL) -> f_agd a1_age
        ^
        |
        Se indica que se recibe el tipo a_atL y se va a utilizar
        en el tipado de la función
```

Indicar que tipo utilizar:

```
h = g @ Int (id @ Int) (GHC.Types.I# 4#)
      ^
      |
      Se indica que se quiere utilizar el tipo Int en g
```

Ejercicio 5

Dado el sistema de clases:

class $A\ a$ (1)
class $(A\ a) \Rightarrow B\ a$ (2)
instance $A\ Bool$ (3)
instance $B\ Bool$ (4)
instance $A\ a \Rightarrow A\ (Maybe\ a)$ (5)
instance $(A\ a, B\ a) \Rightarrow A\ [a]$ (6)

- B Int ⊩ A (Maybe (Maybe Int))

$$\dfrac{\dfrac{\dfrac{}{\text{B Int} \Vdash \text{B Int}}\,(\text{id}) \quad \dfrac{\dfrac{}{\text{class (A a)} \Rightarrow \text{B a}}\,(2)}{\text{class (A Int)} \Rightarrow \text{B Int}}\,[\text{a := Int}]\,(\text{clausura})}{\text{B Int} \Vdash \text{A Int}}\,(\text{super}) \quad \dfrac{\dfrac{}{\text{instance A a} \Rightarrow \text{A (Maybe a)}}\,(5)}{\text{instance A Int} \Rightarrow \text{A (Maybe Int)}}\,[\text{a := Int}]\,(\text{clausura})}{\text{B Int} \Vdash \text{A (Maybe Int)}\,(*)}\,(\text{inst})$$

$$\dfrac{\dfrac{}{\text{B Int} \Vdash \text{A (Maybe Int)}}\,(*) \quad \dfrac{\dfrac{}{\text{instance A a} \Rightarrow \text{A (Maybe a)}}\,(5)}{\text{instance A (Maybe Int)} \Rightarrow \text{A (Maybe (Maybe Int))}}\,[\text{a := Maybe Int}]\,(\text{clausura})}{\text{B Int} \Vdash \text{A (Maybe (Maybe Int))}}\,(\text{inst})$$

- Ø ⊩ A (Maybe [Bool])

$$\dfrac{\dfrac{\dfrac{}{\text{instance (A a, B a)} \Rightarrow \text{A [a]}}\,(6)}{\text{instance (A Bool, B Bool)} \Rightarrow \text{A [Bool]}}\,[\text{a := Bool}]\,(\text{clausura}) \quad \dfrac{\dfrac{}{\text{Ø} \Vdash \text{Ø}}\,(\text{id}) \quad \dfrac{}{\text{instance Ø} \Rightarrow \text{A Bool}}\,(3)}{\text{Ø} \Vdash \text{A Bool}}\,(\text{inst}) \quad \dfrac{\dfrac{}{\text{Ø} \Vdash \text{Ø}}\,(\text{id}) \quad \dfrac{}{\text{instance Ø} \Rightarrow \text{B Bool}}\,(4)}{\text{Ø} \Vdash \text{B Bool}}\,(\text{inst})}{\text{Ø} \Vdash \text{A Bool, B Bool}}\,(\text{uni})}{\text{Ø} \Vdash \text{A [Bool]}\,(*)}\,(\text{inst})$$

$$\dfrac{\dfrac{}{\text{Ø} \Vdash \text{A [Bool]}}\,(*) \quad \dfrac{\dfrac{}{\text{instance A a} \Rightarrow \text{A (Maybe a)}}\,(5)}{\text{instance A [Bool]} \Rightarrow \text{A (Maybe [Bool])}}\,[\text{a := [Bool]}]\,(\text{clausura})}{\text{Ø} \Vdash \text{A (Maybe [Bool])}}\,(\text{inst})$$

Ejercicio 6

La solución se encuentra en el archivo `ej6.hs`.

Ejercicio 7

El código está en el archivo `ej7.hs`.

b)

El tipo SafeList tiene la información de cuando una lista tiene al menos un elemento, por lo tanto, una función que toma como entrada una `SafeList a NonEmpty` solo sabe que la lista tendrá al menos un elemento, en este caso, `safeTail` retorna la cola de una lista no vacía, por lo que no podemos asegurar que la lista retornada tenga algún elemento, para atacar este problema se utiliza Either, a través del cual es posible retornar una lista vacía o una con elementos.