1 2C2024 Recu

1.1 Ejercicio 1

En este ejercicio no se permite utilizar recursión explícita, a menos que se indique lo contrario. El siguiente tipo de datos sirve para representar operadores que realizan operaciones combinadas sobre números enteros. Por simplicidad, modelaremos solamente las sumas y divisiones enteras:

```
data Operador = Sumar Int | DividirPor Int | Secuencia [Operador]
```

`Sumar n` representa la operación que suma `n` a un número entero. `DividirPor n` representa la operación que divide a un entero por `n` (descartando el resto). `Secuencia ops` representa la composición a izquierda de todas las operaciones en `ops`. En otras palabras, representa la operación de aplicar a un número todas las operaciones en `ops` de izquierda a derecha, siendo resultado de cada operación la entrada de la siguiente. Por ejemplo:

```
Secuencia [Sumar 5, DividirPor 2]
```

representa la operación que, dado un entero, le suma 5, y al resultado lo divide por 2.

a) Nos piden foldear la estructura con **foldOperador**

```
-- foldOperador
foldOperador :: (Int -> a) -> (Int -> a) -> ([a] -> a) -> Operador -> a
foldOperador fSuma fDividir fSecuencia operator = case operator of
    Sumar n -> fSuma n
    DividirPor n -> fDividir n
    Secuencia op -> fSecuencia (map rec op)
        where rec = foldOperador fSuma fDividir fSecuencia
```

b) Tenemos que indicar si **falla** cuando hay una división por cero

```
falla :: Operador -> Bool
falla = foldOperador (\n -> False) (\i -> if i == 0 then True else False) (\l -> or l )
```

c) Queremos **aplanar** las subsecuencias para sólo dejar una : Nos recomiendan usar *concatMap* y hacer una auxiliar para usar con ella.

```
aplanar :: Operador -> Operador
aplanar = foldOperador (\o -> Sumar o) (\o -> DividirPor o) (\l -> Secuencia (concatMap listar l))
    where
        listar (Secuencia lista) = concatMap listar lista   -- buscamos aplanar recursivamente
        listar operacion = [operacion]                       -- caso base cuando ya no hay secuencia
```

Básicamente lo que hacemos es, en el caso peludo de la secuencias, agarrar a cada "elemento simple" (sumar / dividir) y lo convertimos en una lista de un elemento . Después el concatMap se encarga de aplicarle esa auxiliar que aplana a cada subsecuencia y unificar todo en una sola lista.

d) No sé si entendí bien el ejemplo, pero la idea es componer de izquierda a derecha y por eso pensé en *foldl*

```
componerTodas ::  [a -> a] -> (a -> a)
componerTodas = foldl (.) id
```

Caso lista vacía devuelve *id*, sino, va componiendo de izquierda a derecha

e) *incompleto*

1.2 Ejercicio 2

Considerar las siguientes definiciones:

```
     const :: a -> b -> a
{C}  const = (\x -> \y -> x)

     head :: [a] -> a
{H}  head (x:xs) = x

     tail :: [a] -> [a]
{T}  tail (x:xs) = xs

     length :: [a] -> Int
{L0} length [] = 0
{L1} length (x:xs) = 1 + length xs

```

```
    null :: [a] -> Bool
{N0}null [] = True
{N1}null (x:xs) = False

    zip :: [a] -> [b] -> [(a,b)]
{Z0}zip [] = const []
{Z1}zip (x:xs) = \ys -> if null ys then [] else (x, head ys) : zip xs (tail ys)
```

a) Nos piden demostrar

$$\forall xs, ys : [a]. \quad length\ (zip\ xs\ ys) = \min(length\ xs)(length\ ys)$$

Vamos a proceder con inducción sobre el tipo **[a]**. Para ello, enunciamos

$\forall x : a, xs, ys : [a]. \quad P(xs) = length\ (zip\ xs\ ys) = \min(length\ xs)(length\ ys)$. Queremos ver que $P(xs) \implies P(x : xs)$

- Caso Base : $P([\,])$

$$\begin{aligned}
P([\,]) &= length\ (zip\ [\,]\ ys) = \min(length\ [\,])(length\ ys)\\
&\overset{\{L0\}}{\equiv} length\ (zip\ [\,]\ ys) = \min(0)(length\ ys)\\
&\overset{\{Z0\}}{\equiv} length\ (const\ [\,]) = \min(0)(length\ ys)\\
&\overset{\{C\}}{\equiv} length\ [\,] = \min(0)(length\ ys)\\
&\overset{\{L0\}}{\equiv} 0 = \min(0)(length\ ys)
\end{aligned}$$

Que da lugar a dos casos, cuando

1. **ys es vacío**

$$\begin{aligned}
&0 = \min(0)(length\ [\,])\\
&\overset{\{L0\}}{\equiv} 0 = \min(0)(0)\\
&\overset{min}{\equiv} 0 = 0\\
&\equiv True
\end{aligned}$$

2. **ys es no vacío**

$$\begin{aligned}
&0 = \min(0)(length\ ys) \text{ en particular length ys } > 0\\
&\overset{min}{\equiv} 0 = 0\\
&\equiv True
\end{aligned}$$

- Paso inductivo. Como dijimos, queremos probar que $P(xs) \implies P(x : xs)$. Asumimos verdadera $P(xs)$ y por tanto, será nuestra **HI**:

$$P(xs) = length\ (zip\ xs\ ys) = \min(length\ xs)(length\ ys)$$

$$\begin{aligned}
P(x : xs) &= length\ (zip\ (x : xs)\ ys) = \min(length\ (x : xs))(length\ ys)\\
&\overset{\{L1\}}{\equiv} length\ (zip\ (x : xs)\ ys) = \min(1 + length\ xs))(length\ ys)\\
&\overset{\{Z1\}}{\equiv} length\ ((\backslash l \to if\ null\ l\ then\ [\,]\ else\ (x,\ head\ l) : zip\ xs\ (tail\ l))\ ys) = \dots\\
&\overset{\beta}{\equiv} length\ (if\ null\ ys\ then\ [\,]\ else\ (x,\ head\ ys) : zip\ xs\ (tail\ ys)) = \dots
\end{aligned}$$

Lo que nos lleva dos casos,

1. Caso **ys vacía**

$$\begin{aligned}
&length\ (if\ null\ [\,]\ then\ [\,]\ else\ (x,\ head\ [\,]) : zip\ xs\ (tail\ [\,])) = \min(1 + length\ xs))(length\ [\,])\\
&\overset{\{N1\}}{\equiv} length([\,]) = \min(1 + length\ xs))(length\ [\,])\\
&\overset{\{L0\}}{\equiv} 0 = \min(1 + length\ xs))(0)\\
&\overset{\{LEMA\}}{\equiv} 0 = 0\\
&\equiv True
\end{aligned}$$

2. Caso **ys no vacía**

$$\begin{aligned}
& length\ (if\ null\ (y : ys)\ then\ [\,]\ else\ (x,\ head\ (y : ys)) : zip\ xs\ (tail\ (y : ys))) = \ldots \\
& \equiv length\ ((x,\ head\ (y : ys)) : zip\ xs\ (tail\ (y : ys))) = \ldots \\
& \overset{\{H\}\{T\}}{\equiv} length\ ((x,\ y) : zip\ xs\ ys) = \ldots \\
& \overset{\{L1\}}{\equiv} 1 + length\ (zip\ xs\ ys) = \min(1 + length\ xs)(length\ (y : ys)) \\
& \overset{\{L1\}}{\equiv} 1 + length\ (zip\ xs\ ys) = \min(1 + length\ xs)(1 + length\ ys) \\
& \overset{\{HI\}}{\equiv} True \text{ ya que el 1 se suma en todos los subtérminos de un lado y del otro de la igualdad}
\end{aligned}$$

b) Nos piden demostrar el siguiente teorema. Vale usar principios clásicos

$$(\tau \implies (\sigma \wedge \rho)) \vee (\rho \implies (\sigma \implies \tau))$$

EL truco para este ejercicio es que sabemos que :

- $Falso \implies \star\ es\ Verdadero$
- $\star \implies \star\ es\ Verdadero$

Con esto en mente, elegimos que $\star = \tau$ y entonces ...

$$\dfrac{\dfrac{}{\vdash \tau \vee \neg\tau}\,\text{LEM} \qquad \dfrac{\dfrac{\dfrac{}{\tau, \rho, \sigma \vdash \tau}\,\text{ax}}{\tau, \rho \vdash \sigma \implies \tau}\,\Rightarrow_i}{\tau \vdash (\tau \implies (\sigma \wedge \rho)) \vee (\rho \implies (\sigma \implies \tau))}\,\vee_{i_2} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\neg\tau, \tau \vdash \tau}\,ax \qquad \dfrac{}{\neg\tau, \tau \vdash \neg\tau}\,ax}{\neg\tau, \tau \vdash \bot}\,\neg_e}{\neg\tau, \tau \vdash \sigma \wedge \rho}\,\bot_e}{\neg\tau \vdash \tau \implies (\sigma \wedge \rho)}\,\Rightarrow_i}{\neg\tau \vdash (\tau \implies (\sigma \wedge \rho)) \vee (\rho \implies (\sigma \implies \tau))}\,\vee_{i_1}}{\vdash (\tau \implies (\sigma \wedge \rho)) \vee (\rho \implies (\sigma \implies \tau))}$$

1.3 Ejercicio 3

Se extenderán los tipos y términos de la siguiente manera:

$$\begin{aligned}
\tau &::= \cdots \mid \text{Cola } \tau \\
M &::= \cdots \mid \langle\rangle_\tau \mid M \bullet M \mid \text{recr } M \triangleright \langle\rangle \rightsquigarrow M; r, c \bullet x \rightsquigarrow M
\end{aligned}$$

a) Introducimos las reglas de tipado

$$\dfrac{}{\Gamma \vdash \langle\rangle_\tau : Cola_\tau}\,\text{T-COLAEMPTY}$$

$$\dfrac{\Gamma \vdash M_1 : Cola_\tau \qquad \Gamma \vdash M_2 : \tau}{\Gamma \vdash M_1 \bullet M_2 : Cola_\tau}\,\text{T-ENCOLAR}$$

$$\dfrac{\Gamma \vdash M_1 : Cola_\tau \qquad \Gamma \vdash M_2 : \sigma \qquad \Gamma, r : \sigma, c : Cola_\tau, x : \tau \vdash M_3 : \sigma}{\Gamma \vdash \text{recr } M_1 \triangleright \langle\rangle \rightsquigarrow M_2; r, c \bullet x \rightsquigarrow M_3 : \sigma}\,\text{T-RECR}$$

b) demostrar validez del siguiente juicio

$$\dfrac{\dfrac{\dfrac{}{c : Cola_{Nat} \vdash c : Cola_{Nat}}\,\text{ax} \qquad \dfrac{}{c : Cola_{Nat} \vdash \langle\rangle_{Bool} : Cola_{Bool}}\,\text{T-C.E} \qquad \dfrac{\dfrac{}{\Gamma' \vdash r : Cola_{Bool}}\,\text{ax} \qquad \dfrac{}{\Gamma' \vdash True : Bool}\,\text{ax}}{c : Cola_{Nat}, r : Cola_{Bool}, y : Cola_{Nat}, x : Nat \vdash r \bullet True : Cola_{Bool}}\,\text{T-ENCOLAR}}{c : Cola_{Nat} \vdash \text{recr } c \triangleright \langle\rangle \rightsquigarrow \langle\rangle_{Bool}; r, y \bullet x \rightsquigarrow r \bullet True : Cola_{Bool}}\,\text{T-RECR}}{\vdash \lambda c : Cola_{Nat}.\text{recr } c \triangleright \langle\rangle \rightsquigarrow \langle\rangle_{Bool}; r, y \bullet x \rightsquigarrow r \bullet True : (Cola_{Nat} \to Cola_{Bool})}\,\text{T-ABS}$$

c) Conjunto de valores y reglas de cómputo

$$V ::= \ldots \mid \langle\rangle_\tau \mid V \bullet V$$

Reglas de cómputo

$recr\ \langle\rangle_\tau \triangleright \langle\rangle \rightsquigarrow M_2; r, c \bullet x \rightsquigarrow M_3 \to M_2$ `E-RECREMPTY`

$recr\ V_1 \bullet V_2 \triangleright \langle\rangle \rightsquigarrow M_2; r, c \bullet x \rightsquigarrow M_3 \to M_3\{r := recr\ V_1 \triangleright \langle\rangle \rightsquigarrow M_2; r, c \bullet x \rightsquigarrow M_3\}\{c := V_1\}\{x := V_2\}$ `E-RECR1`

Reglas de congruencia $\mathbf{M} \to \mathbf{M'}$

$$M \bullet V \to M' \bullet V \quad \texttt{E-ENCOLAR1}$$
$$V \bullet M \to V \bullet M' \quad \texttt{E-ENCOLAR2}$$
$$recr\ M \triangleright \langle\rangle \rightsquigarrow M_2; r, c \bullet x \rightsquigarrow M_3 \to recr\ M' \triangleright \langle\rangle \rightsquigarrow M_2; r, c \bullet x \rightsquigarrow M_3 \quad \texttt{E-RECR2}$$

- Reducir :

$$recr\ \langle\rangle_{Nat}\ \bullet zero \bullet \underline{1} \triangleright \langle\rangle \rightsquigarrow \langle\rangle_{Nat};\ r, c \bullet x \rightsquigarrow if\ isZero(x)\ then\ c\ else\ r \bullet x$$
$$\overset{E-RECR1}{\to} if\ isZero(\underline{1})\ then\ (\langle\rangle_{Nat}\ \bullet zero)\ else\ (recr\ \langle\rangle_{Nat}\ \bullet zero...) \bullet \underline{1}$$
$$\overset{E-ISZEROSUCC}{\to} (recr\ \langle\rangle_{Nat}\ \bullet zero \triangleright \langle\rangle \rightsquigarrow \langle\rangle_{Nat};\ r, c \bullet x \rightsquigarrow if\ isZero(x)\ then\ c\ else\ r \bullet x) \bullet \underline{1}$$
$$\overset{E-RECR1}{\to} (if\ isZero(zero)\ then\ \langle\rangle_{Nat}\ else\ r \bullet x) \bullet \underline{1}$$
$$\overset{E-ISZEROZERO}{\to} \langle\rangle_{Nat} \bullet \underline{1}$$