Laboratorio - Parcial 1: Programación Funcional en Haskell

Generales

- Para codear el parcial solo podrán utilizar algún editor de texto plano (sin asistencia) tales como gedit, vim, emacs. No utilizar teléfonos durante el examen. Por favor, ser honestos intelectualmente y no poner al docente en una situación incómoda.🙏
- Pueden utilizar sus propias laptops o alguna máquina del laboratorio que tenga instalado todo lo necesario para el lab1 (ghc, ghci, gloss, etc).
- Pueden tener el código fuente del laboratorio 1 para consulta pero NO deben entregarlo, solo el código fuente del examen deben submitear.
- **NO compila => Reprobado**
- En el archivo Main.hs están definidos los distintos testing para cada uno de los incisos del examen parcial. Por favor, leerlo detenidamente e ir comentando o descomentando los bloques de testing a medida que se van implementando sus respectivas funciones.

Ejercicio 1. Sintaxis

Dado el siguiente sub-lenguaje del laboratorio 1:

Dibujo `a :=`

```
    | Basica a
    | Rotar <Dibujo a>
    | Apilar <Float> <Float> <Dibujo a> <Dibujo a>
    | Encimar <Dibujo a> <Dibujo a>
```

En el archivo `Dibujo.hs` realizar los siguientes incisos:

a) Definir la función toBool :: Dibujo (Int,Int) -> Dibujo Bool que transforma solamente las figuras basicas “Basica (x,y)” en “Basica True” sii “x es múltiplo de y ó viceversa”, utilizando pattern matching en la estructura de la expresión.

Ejemplos:

```
toBool (Basica (2,4)) = Basica (True)
toBool (Basica (3,5)) = Basica (False)
toBool (Rotar (Basica (3,5))) = Rotar (Basica (False))
toBool (Encimar (Basica (2,4)  Basica (3,5)) = (Encimar
(Basica True) (Basica False))
```

Para compilar y correr en la terminal el testing de este inciso:
$ ghc -o Main Main.hs
$ ./Main 1a

```
(base) fbustos@fbustos-CX-Infinito:~/Documents/cursos/famaf/paradigmas/paradigmas2025/parcial1/solved$ ./Main 1a
TESTING FUNCION TOBOOL:
Ok
Ok
Ok
Ok
Ok
Ok
(base) fbustos@fbustos-CX-Infinito:~/Documents/cursos/famaf/paradigmas/paradigmas2025/parcial1/solved$
```

b) Redefinir, sin utilizar pattern matching, la función anterior toBool2 :: Dibujo (Int,Int) -> Bool a través de la función **mapDib** ya implementada por la cátedra.

Para compilar y correr en la terminal el testing de este inciso:
$ ghc -o Main Main.hs
$ ./Main 1b

```
(base) fbustos@fbustos-CX-Infinito:~/Documents/cursos/famaf/paradigmas/paradigmas2025/parcial1/solved$ ./Main 1b
TESTING FUNCION TOBOOL2:
Ok
Ok
Ok
Ok
Ok
Ok
(base) fbustos@fbustos-CX-Infinito:~/Documents/cursos/famaf/paradigmas/paradigmas2025/parcial1/solved$
```

c) Definir la función profundidad :: Dibujo a -> Int que devuelve la profundidad una expresión, utilizando pattern matching en la estructura de la expresión. Recordar que la profundidad de una expresión es igual a la profundidad del árbol de parseo de la expresión.

Ejemplos:
profundidad (Basica (2,4)) = 1
profundidad (Rotar (Basica (2,4))) = 2
profundidad (Rotar (Rotar (Basica (2,4)))) = 3
profundidad (Apilar 1 1 (Basica (3,4)) (Rotar (Rotar (Basica (2,4))))) = 4

Para compilar y correr en la terminal el testing de este inciso:
$ ghc -o Main Main.hs
$ ./Main 1c

```
(base) fbustos@fbustos-CX-Infinito:~/Documents/cursos/famaf/paradigmas/paradigmas2025/parcial1/solved$ ./Main 1c
TESTING FUNCION PROFUNDIDAD:
Ok
Ok
Ok
Ok
Ok
(base) fbustos@fbustos-CX-Infinito:~/Documents/cursos/famaf/paradigmas/paradigmas2025/parcial1/solved$
```

d) Redefinir, sin utilizar pattern matching, la función anterior profundidad2 :: Dibujo a -> Int a través de la función **foldDib** ya implementada por la cátedra.

Para compilar y correr en la terminal el testing de este inciso:
$ ghc -o Main Main.hs
$ ./Main 1d

```
(base) fbustos@fbustos-CX-Infinito:~/Documents/cursos/famaf/paradigmas/paradigmas2025/parcial1/solved$ ./Main 1d
TESTING FUNCION PROFUNDIDAD2:
Ok
Ok
Ok
Ok
Ok
(base) fbustos@fbustos-CX-Infinito:~/Documents/cursos/famaf/paradigmas/paradigmas2025/parcial1/solved$
```

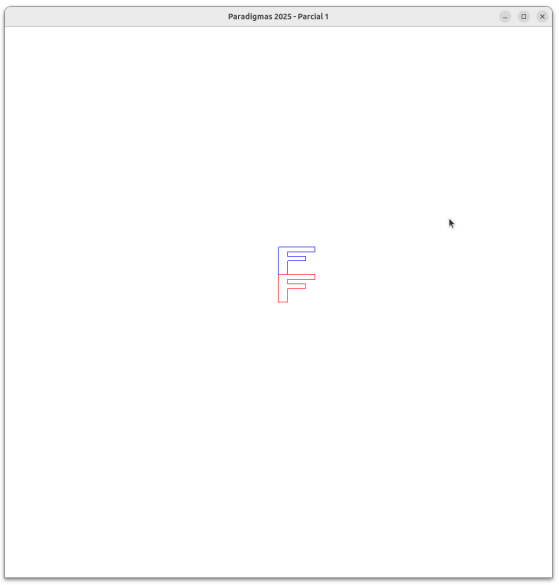
Ejercicio 2. Semántica (Interpretación)

Interpretaremos expresiones de **Dibujo Bool**, donde (Basica True) sera interpretada por una “efe de color azul” y (Basica False) sera intepretada como una “efe de color rojo”. El resto de los operadores (rotar, apilar, encimar) será interpretado de igual manera que en el laboratorio y su función de interpretación ya está provista por la cátedra.

En el archivo `Interp.hs` realizar los siguientes incisos:

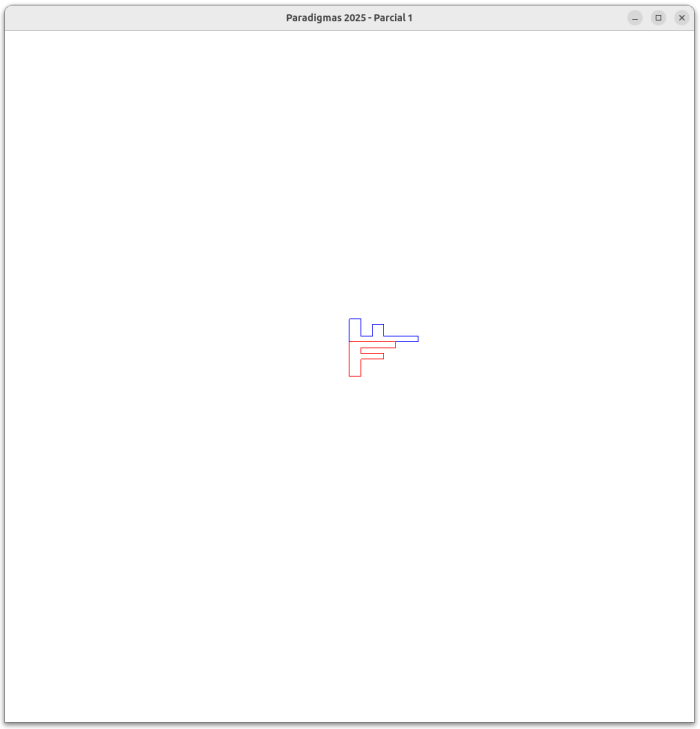
a) Definir la función interp_basica :: Bool -> ImagenFlotante que dada un valor booleano me devuelve “efe de color rojo” ó “efe de color azul” como dijimos arriba.

Para compilar y correr en la terminal el testing de este inciso:
$ ghc -o Main Main.hs
$ ./Main 2a



b) Definir la función interp:: Dibujo Bool -> ImagenFlotante que interpreta una expresión de Dibujo Bool, haciendo pattern matching en la estructura de la expresión y llamando al intérprete de cada constructor según corresponda.

Para compilar y correr en la terminal el testing de este inciso:
$ ghc -o Main Main.hs
$ ./Main 2b



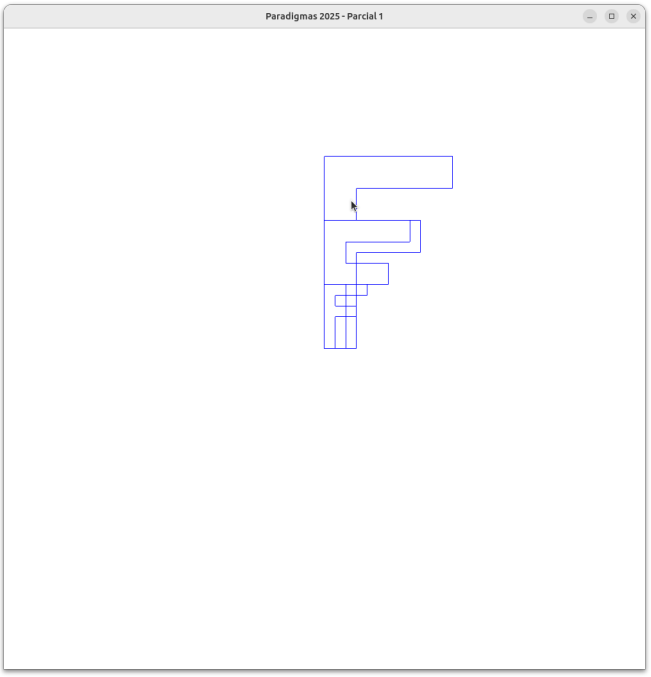
Ejercicio 3. Extensión del sub-lenguaje

a) Definir un nuevo operador del lenguaje “Resize” que toma dos argumentos, un entero (factor de redimensionamiento) y un “Dibujo a”. Este operador redimensiona el dibujo en su width y height, según el factor de redimensionamiento, pero sin modificar su desplazamiento al origen.

Ejemplos:
Resize 1 (Basica True) no altera el tamaño de la imagen
Resize 2 (Basica True) duplica el tamaño de la imagen
Resize 3 (Basica True) triplica el tamaño de la imagen
Resize 4 (Basica True) cuadruplica el tamaño de la imagen
….

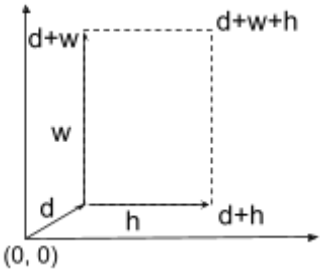
Y luego, agregar el nuevo pattern matching correspondiente al nuevo operador del lenguaje a todas las funciones definidas en Dibujo.hs e Interp.hs de tal manera que **el nuevo constructor "Resize" quede totalmente implementado**.

Para compilar y correr en la terminal el testing (semántico) de este inciso:
$ ghc -o Main Main.hs
$ ./Main 3



Recordar que:

d = vector de desplazamiento del origen
w = vector width
h = vector height



Entrega

Comprimir el directorio raíz de su examen en un archivo "tar.xz" y solo con los .hs (evitar binarios o archivos objetos) con el siguiente formato "Apellido_Nombre.tar.xz" y adjuntar el comprimido en el formulario, llenar todos los campos del mismo y submitear.