FUNDAMENTOS DE COMPUTACIÓN
ENTREGABLE 3
NÚMEROS RACIONALES

Este trabajo tiene un puntaje de 4 puntos, y debe ser realizado en forma INDIVIDUAL, y SIN ASISTENCIA de herramientas de Inteligencia Artificial. Se debe subir a Aulas antes del día 26/4/24 a las 21:00 hs.

Racionales como pares

Se define el tipo `Q` de los números racionales positivos, representados como pares de naturales:

`data Q where { R::N -> N -> Q } deriving Show`

Así, por ejemplo el número racional $\frac{1}{3}$ se representará con la expresión `R (S O) (S(S(S O)))`.

Esta no es la mejor definición, ya que no estamos evitando que el denominador sea cero. Además existen racionales que son iguales, pero tienen diferente representación, como $\frac{1}{2}$, $\frac{2}{4}$, $\frac{4}{8}$, ...
De todos modos hay funciones interesantes que podremos definir con este tipo.

Funciones a implementar

En este trabajo entregable se pide definir las siguientes funciones para los números Racionales definidos arriba. Se deben utilizar las funciones definidas para los Naturales en el Laboratorio 2 que consideren necesarias, las cuales deberán ser importadas de un archivo llamado `Naturales.hs` (leer instrucciones al final):

1) `num::Q -> N`, que devuelva el numerador de un número racional.
   Ejemplo: `num (R unos tres) = S O`

2) `den::Q -> N`, que devuelva el denominador de un número racional.
   Ejemplo: `den (R cinco dos) = S(S O)`

3) Implemente la instancia de `Eq` para `Q`, o sea: defina la igualdad de Racionales, teniendo en cuenta que un mismo número puede tener diferentes representaciones.
   En este ejercicio y los subsiguientes puede asumir siempre que los racionales que se reciben tienen denominador distinto de cero.
   Ejemplos:
   ```
   R uno dos == R dos cuatro = True
   R uno dos == R dos dos = False
   R dos tres /= R cuatro uno = True
   ```

4) Implemente la instancia de `Ord` para `Q`, o sea: defina la función `(<=)` para los Racionales.
   Ejemplos:
   ```
   R uno tres <= R uno dos = True
   R dos cinco > R uno dos = False
   R dos tres < R uno dos = False
   R dos tres >= R cuatro seis = True
   ```

5) Implemente la instancia de `Num` para `Q`, o sea: defina las funciones `(+)`, `(*)` y `(-)` para los Racionales.
   Ejemplos:
```
R uno tres + R uno dos == R cinco seis = True
R dos tres + R 0 dos == R cuatro seis = True
R dos tres * R uno dos == R dos seis = True
R tres dos * R uno dos == R tres cuatro = True
R uno tres - R uno dos == R 0 seis = True
R uno dos - R uno tres == R uno seis = True
```

6) Defina la función `sumFracc :: N -> Q`, tal que:

$$\texttt{sumFracc n} = \sum_{i=1}^{\texttt{n}} \frac{1}{i}$$

   Si `n` es cero, se deberá devolver como resultado `0`.

   Ejemplos: `sumFracc dos == R tres dos = True`, ya que $\frac{1}{1} + \frac{1}{2} = \frac{3}{2}$
```
sumFracc tres == R once seis = True
```

Entregables

- El trabajo deberá realizarse en forma individual.
- La entrega deberá realizarse antes del **26/4/2024 a las 21:00 hs.**
- Deberan subirse a Aulas dos archivos Haskell:
  - `Naturales.hs`, con las funciones del laboratorio 2 que hayan definido para `N` y que sean necesarias para la solución (como las instancias de Eq, Ord, Num, etc. para los Naturales), y
  - `Racionales.hs`, con el código fuente de la solución.
- En Aulas se encuentra el archivo `Racionales.hs` con las funciones que deben implementarse. Para facilitar la corrección deberá usarse este último como template.
- No se corregirán archivos que no compilen, por lo que recomendamos comentar el código que no compile y dejar como `undefined` las funciones no implementadas.
- Se utilizarán herramientas para la detección de copias y de uso de herramientas de IA, por lo que recomendamos NO COMPARTIR CÓDIGO, ya que las copias serán penalizadas.