

# Programación para la Inteligencia Artificial

## Haskell - Práctica 4

### Funciones, tipos y clases de tipos

El tipo *Racional* se puede representar con dos enteros, de forma que uno de ellos represente al numerador y el otro al denominador. Pueden definirse las siguientes operaciones:

- *simplificaRac*: Obtiene la representación canónica de un racional. Por ejemplo:

```
simplificaRac 6/12 = 1/2
simplificaRac 6/-12 = -1/2
simplificaRac -6/-12 = 1/2
simplificaRac -6/12 = -1/2
```

- *multRac*: Multiplica dos racionales. Por ejemplo:

```
multRac 1/4 4/5 = 1/5
```

- *divRac*: Divide dos racionales. Por ejemplo:

```
divRac 1/4 4/5 = 5/16
```

- *sumRac*: Suma dos racionales. Por ejemplo:

```
sumRac 1/4 4/5 = 21/20
```

- *resRac*: Resta dos racionales. Por ejemplo:

```
resRac 1/4 4/5 = -11/20
```

Define en Haskell el tipo *Racional* de dos formas diferentes:

1. Como renombramiento de un par de enteros (`Int,Int`). Definir las funciones anteriores<sup>1</sup>. Definir, además, una función *muestraRac* que convierta un *Racional* en su cadena correspondiente. Por ejemplo:

```
muestraRac (6,12) = "1/2"
muestraRac (6,6)  = "1"
muestraRac (multRac (1,2) (2,3)) = "1/3"
```

2. Como un nuevo tipo de datos, por ejemplo:

```
data Racional = Rac Integer Integer
```

Definir las funciones anteriores<sup>1</sup>. Definir, además, *Racional* como una instancia de `Show`, de forma que la función `show` muestre la forma simplificada obtenida mediante la función `simplificaRac` definida al principio.

3. Completar el ejercicio anterior declarando el tipo *Racional* como una instancia de la clase `Num`, redefiniendo las operaciones `(*)`, `(+)`, `(-)`, `negate` (que modifica el signo a un número racional), `fromInteger` (que convierte un entero en racional), `signum` y `abs`.

---

<sup>1</sup>Pueden usarse las funciones predefinidas `signum`, que devuelve `-1` si el número es negativo y `1` en otro caso, `abs`, que devuelve el valor absoluto de un número, `gcd`, que devuelve el *máximo común divisor* de dos números y `lcm` que devuelve el *mínimo común múltiplo*.