

Ficha 10

Programação Funcional

2015/16

1. Considere o seguinte tipo de dados para representar fracções

```
data Frac = F Integer Integer
```

- (a) Defina a função `normaliza :: Frac -> Frac`, que dada uma fracção calcula uma fracção equivalente, irredutível, e com o denominador positivo.

Por exemplo: `normaliza (F (-33) (-51)) = (F 11 17)`

`normaliza (F 50 (-5)) = (F (-10) 1)`

Sugestão: defina primeiro a função `mdc :: Integer -> Integer -> Integer` que calcula o máximo divisor comum entre dois números baseada na seguinte propriedade (atribuída a *Euclides*): $\text{mdc } x \ y = \text{mdc } (x + y) \ y = \text{mdc } x \ (y + x)$

- (b) Defina `Frac` como instância da classe `Eq`.
(c) Defina `Frac` como instância da classe `Ord`.
(d) Defina `Frac` como instância da classe `Show`, de forma a que cada fracção seja apresentada por *(numerador/denominador)*.
(e) Defina `Frac` como instância da classe `Num`. Relembre que a classe `Num` tem a seguinte definição

```
class (Eq a, Show a) => Num a where
  (+), (*), (-) :: a -> a -> a
  negate, abs, signum :: a -> a
  fromInteger :: Integer -> a
```

- (f) Defina uma função que, dada uma fracção `f` e uma lista de fracções `l`, selecciona de `l` os elementos que são maiores do que o dobro de `f`.

2. Relembre o tipo definido na Ficha 9 para representar expressões inteiras. Uma possível generalização desse tipo de dados, será considerar expressões cujas constantes são de um qualquer tipo numérico (i.e., da classe `Num`).

```
data Exp a = Const a
           | Simetrico (Exp a)
           | Mais (Exp a) (Exp a)
           | Menos (Exp a) (Exp a)
           | Mult (Exp a) (Exp a)
```

- (a) Declare `Exp a` como uma instância de `Show`.
(b) Declare `Exp a` como uma instância de `Eq`.
(c) Declare `Exp a` como instância da classe `Num`.

3. Relembre o exercício da Ficha 7 sobre contas bancárias, com a seguinte declaração de tipos

```
data Movimento = Credito Float | Debito Float
data Data = D Int Int Int      -- Dia Mes Ano
data Extracto = Ext Float [(Data, String, Movimento)]
```

- (a) Defina `Data` como instância da classe `Ord`.
- (b) Defina `Data` como instância da classe `Show`.
- (c) Defina a função `ordena :: Extracto -> Extracto`, que transforma um extracto de modo a que a lista de movimentos apareça ordenada por ordem crescente de data.
- (d) Defina `Extracto` como instância da classe `Show`, de forma a que a apresentação do extracto seja por ordem de data do movimento com o seguinte, e com o seguinte aspecto

Saldo anterior: 300

Data	Descricao	Credito	Debito

2010/4/5	DEPOSITO	2000	
2010/8/10	COMPRA		37,5
2010/9/1	LEV		60
2011/1/7	JUROS	100	
2011/1/22	ANUIDADE		8

Saldo actual: 2294,5