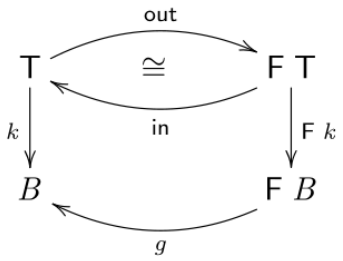
CP - Ficha 7

> O quadro abaixo representa a **propriedade universal** que define o combinador **catamorfismo**, com duas instâncias — números naturais $\mathbb{N}_0$ e listas e listas finitas A^* , onde $\hat{f}$ abrevia $\mathrm{uncurry}\, f$.

Catamorfismo *(Catamorphism)*:



$$k = (\!|\, g \,|\!) \;\Leftrightarrow\; k \cdot \mathsf{in} = g \cdot \mathsf{F}\ k \qquad \text{(F1)}$$

Listas *(Lists)*:

$$\begin{cases} \mathsf{T} = A^* \\ \begin{cases} \mathsf{in} = [\mathsf{nil}\ , \mathsf{cons}] \\ \mathsf{nil}\ \_ = [\,] \\ \mathsf{cons}\ (h, t) = h : t \end{cases} \\ \begin{cases} \mathsf{F}\ X = 1 + A \times X \\ \mathsf{F}\ f = id + id \times f \end{cases} \end{cases} \qquad \mathsf{foldr}\ f\ i = (\!|\, [\underline{i}\ , \widehat{f}] \,|\!)$$

Números naturais *(Natural numbers)*:

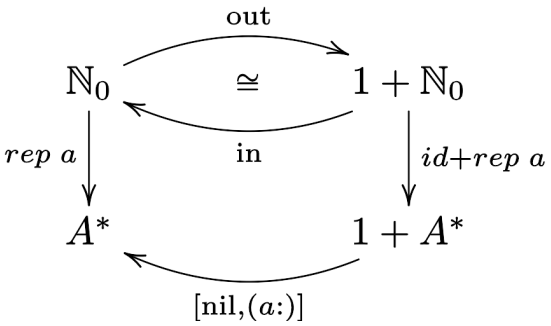
$$\begin{cases} \mathsf{T} = \mathbb{N}_0 \\ \begin{cases} \mathsf{in}_{\mathbb{N}_0} = [\underline{0}\ , \mathsf{succ}] \\ \mathsf{succ}\quad n = n + 1 \end{cases} \\ \begin{cases} \mathsf{F}\ X = 1 + X \\ \mathsf{F}\ f = id + f \end{cases} \end{cases} \qquad \mathsf{for}\ b\ i = (\!|\, [\underline{i}\ , b] \,|\!)$$

Exercício 1

> Fazendo $\mathrm{T} = \mathbb{N}_0$, codifique — recorrendo à biblioteca Cp.hs e à definição de out feita numa ficha anterior — o combinador:
>
> $$(\!|g|\!) = g \cdot (id + (\!|g|\!)) \cdot \mathrm{out} \qquad \text{(F2)}$$
>
> De seguida implemente e teste a seguinte função:
>
> $$rep\ a = (\!|[\mathrm{nil}, (a:)]|\!) \qquad \text{(F3)}$$
>
> O que faz ela?

Resolução 1

```haskell
{-# LANGUAGE NPlusKPatterns #-}

import Cp (i1, i2, (-|-), nil)

-- Definição de out para números naturais
out 0     = i1 ()
out (n+1) = i2 n

-- Definição de functor para números naturais
fF f = id -|- f

-- Definição do catamorfismo para números naturais
cata g = g . fF (cata g) . out

-- Definição de rep
rep a = cata (either nil (a:))



-- Testes da função rep
ghci> rep "abc" 0
[]
ghci> rep "abc" 1
["abc"]
ghci> rep True 2
[True, True]
```



$$rep : \mathrm{A} \to \mathbb{N}_0 \to \mathrm{A}^*$$
$$rep\ a = \mathrm{for}\ (a:)\ [\,] \qquad (\text{Def. for, Def. nil})$$

A função $rep\ a\ n$ cria uma lista com n elementos iguais a a.

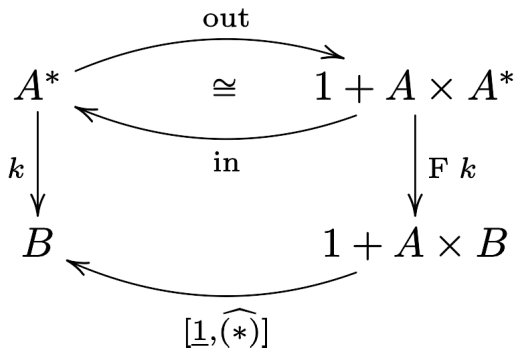
Exercício 2

> Identifique como catamorfismos de listas ($k = (\!|g|\!)$) as funções seguintes, indicando o gene g para cada caso (apoie a sua resolução com diagramas):
>
> a) k é a função que multiplica todos os elementos de uma lista.
>
> b) $k = \mathrm{reverse}$
>
> c) $k = \mathrm{concat}$
>
> d) k é a função $\mathrm{map}\ f$, para um dado $f : A \to B$.
>
> e) k é a função que calcula o máximo de uma lista de números naturais $(\mathbb{N}_0{}^*)$.
>
> f) $k = \mathrm{filter}\ p$ onde:

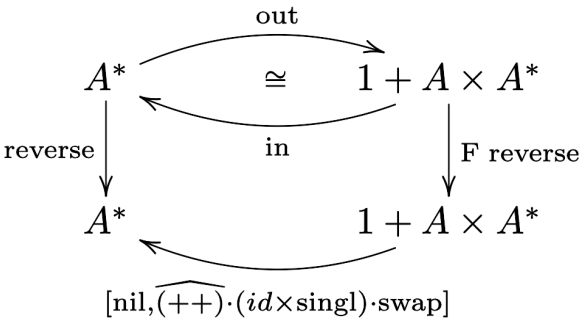
```haskell
filter p [] = []
filter p (h:t) = x ++ filter p t
  where x = if (p h) then [h] else []
```

Resolução 2

a)

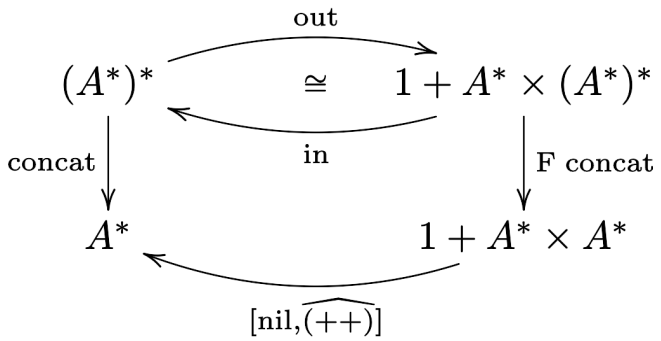


b)

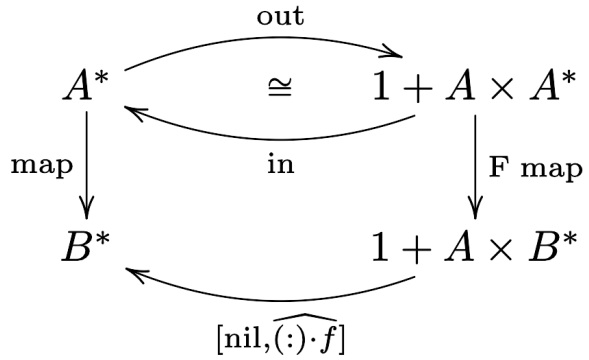


NB: $g_2 = \widehat{(++)} \cdot (id \times \mathrm{singl}) \cdot \mathrm{swap} \quad \equiv \quad g_2\ (a, b) = b +\!\!+ [a]$

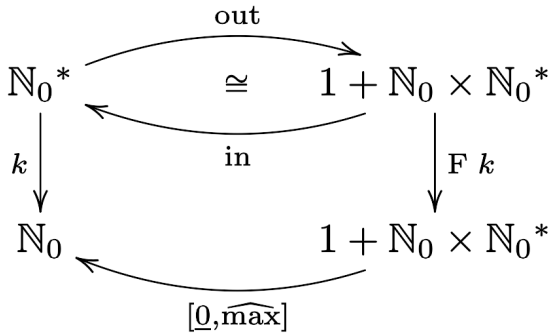
c)


out
$(A^*)^*$
$\cong$
$1 + A^* \times (A^*)^*$
in
concat
F concat
A^*
$1 + A^* \times A^*$
$[\mathrm{nil}, \widehat{(++)}]$

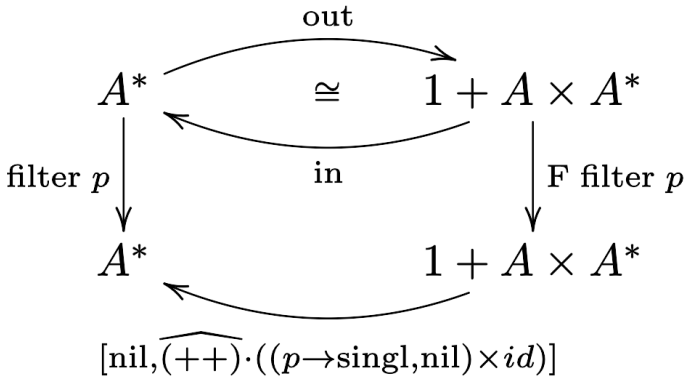
d)


out
A^*
$\cong$
$1 + A \times A^*$
in
map
F map
B^*
$1 + A \times B^*$
$[\mathrm{nil}, \widehat{(:) \cdot f}]$

e)


out
$\mathbb{N}_0{}^*$
$\cong$
$1 + \mathbb{N}_0 \times \mathbb{N}_0{}^*$
in
k
F k
$\mathbb{N}_0$
$1 + \mathbb{N}_0 \times \mathbb{N}_0{}^*$
$[\underline{0}, \widehat{\mathrm{max}}]$

f)


out
A^*
$\cong$
$1 + A \times A^*$
in
filter p
F filter p
A^*
$1 + A \times A^*$
$[\mathrm{nil}, \widehat{(++)} \cdot ((p \to \mathrm{singl}, \mathrm{nil}) \times id)]$

Exercício 3

> A função seguinte, em Haskell
>
> ```haskell
> sumprod a [] = 0
> sumprod a (h:t) = a * h + sumprod a t
> ```
>
> é o catamorfismo de listas
>
> $$\text{sumprod}\ a = (\!|[\text{zero}, \text{add} \cdot ((a*) \times id)]|\!) \tag{F4}$$
>
> onde $\text{zero} = \underline{0}$ e $\text{add}\ (x, y) = x + y$. Como exemplo de aplicação da propriedade de **fusão-cata** para listas, demonstre a igualdade
>
> $$\text{sumprod}\ a = (a*) \cdot \text{sum} \tag{F5}$$
>
> onde $\text{sum} = (\!|[\text{zero}, \text{add}]|\!)$.
> **NB:** não ignore propriedades elementares da aritmética que lhe possam ser úteis.

$$
\begin{aligned}
&\text{sumprod}\ a = (\!|[\text{zero}, \text{add} \cdot ((a*) \times id)]|\!) \\
&\equiv && (\text{Def. sum, F5}) \\
&(a*) \cdot (\!|[\text{zero}, \text{add}]|\!) = (\!|[\text{zero}, \text{add} \cdot ((a*) \times id)]|\!) \\
&\Leftarrow && (\text{49: Fusão-cata, F}\ (a*) = id + id \times (a*)\ ) \\
&(a*) \cdot [\text{zero}, \text{add}] = [\text{zero}, \text{add} \cdot ((a*) \times id)] \cdot (id + id \times (a*)) \\
&\equiv && (\text{20: Fusão-+, 22: Absorção-+, 27: Eq-x}) \\
&\begin{cases} (a*) \cdot \text{zero} = \text{zero} \cdot id \\ (a*) \cdot \text{add} = \text{add} \cdot ((a*) \times id) \cdot (id \times (a*)) \end{cases} \\
&\equiv && (\text{1: Natural-id, Def. zero, 72: Ig. Ext., 73: Def-comp, 75: Def. const}) \\
&\begin{cases} a * 0 = 0 \\ a * (\text{add}\ (x, y)) = \text{add} \cdot ((a*) \times id) \cdot (id \times (a*))\ (x, y) \end{cases} \\
&\equiv && (\text{Def. add}\ (2\times), \text{73: Def-comp, 78: Def-}\times\ (2\times), \text{74: Def. id}\ (2\times)) \\
&\begin{cases} a * 0 = 0 \\ a * (x + y) = a * x + a * y \end{cases} \\
&\equiv && (\text{Elem. absorvente e distributividade da multiplicação}) \\
&\text{True}
\end{aligned}
$$

Exercício 4

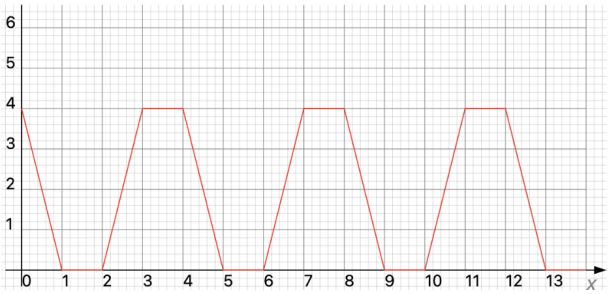
> A função $\text{foldr}\ \overline{\pi_2}\ i$ é necessariamente uma função constante. Qual? Justifique com o respectivo cálculo.

$$
\begin{aligned}
&\underline{k} = \text{foldr}\ \overline{\pi_2}\ i \\
&\equiv && (\text{Def. foldr}) \\
&\underline{k} = (\!|[\underline{i}, \widehat{\overline{\pi_2}}]|\!) \\
&\equiv && (\text{Iso. curry/uncurry, 46: Universal-cata}) \\
&\underline{k} \cdot \text{in} = [\underline{i}, \pi_2] \cdot (id + id \times \underline{k}) \\
&\equiv && (\text{Def. in, 20: Fusão-}+\text{, 22: Absorção-}+) \\
&[\underline{k} \cdot \text{nil}, \underline{k} \cdot \text{cons}] = [\underline{i} \cdot id, \pi_2 \cdot (id \times \underline{k})] \\
&\equiv && (\text{3: Fusão-const, 27: Eq-}+) \\
&\begin{cases} \underline{k} = \underline{i} \\ \underline{k} = \pi_2 \cdot (id \times \underline{k}) \end{cases} \\
&\equiv && (\text{12: Natural-}\pi_2\text{, 3: Fusão-const}) \\
&\begin{cases} \underline{k} = \underline{i} \\ \underline{k} = \underline{k} \end{cases}
\end{aligned}
$$

$$\text{Logo,} \quad \text{foldr}\ \overline{\pi_2}\ i = \underline{i}$$

Exercício 5

A figura representa a função $\pi_1 \cdot aux$, para aux definida ao lado:




```
aux = for loop (4, -2)
  where loop (a, b) = (2 + b, 2 - a)
```

Partindo da definição do combinador $\text{for}\ b\ i = (\!|[\underline{i}, b]|\!)$, para $\mathrm{F} = id + f$ e $\text{in} = [\underline{0}, \text{succ}]$, resolva em ordem a f e g a equação

$$\langle f, g \rangle = aux$$

por aplicação da lei de recursividade mútua, entregando as definições de f e g em notação *pointwise*.

$$\begin{aligned}
&\text{loop}\ (a, b) = (2 + b, 2 - a) \\
&\equiv && (78\text{: Def-}\times) \\
&\text{loop}\ (a, b) = ((2+) \times (2-))\ (b, a) \\
&\equiv && (\text{Def. pointwise swap, 72: Ig. Ext.}) \\
&\text{loop} = ((2+) \times (2-)) \cdot \text{swap} \\
&\equiv && (\text{Def. pointfree swap}) \\
&\text{loop} = ((2+) \times (2-)) \cdot \langle \pi_2, \pi_1 \rangle \\
&\equiv && (11\text{: Absorção-}\times) \\
&\text{loop} = \langle (2+) \cdot \pi_2, (2-) \cdot \pi_1 \rangle
\end{aligned}$$

$$
\begin{aligned}
&\langle f, g\rangle = aux \\
&\equiv && (\text{Def. } aux) \\
&\langle f, g\rangle = \text{for loop } (4, -2) \\
&\equiv && (\text{Def. for, Def. loop, } \underline{(a, b)} = \langle \underline{a}, \underline{b}\rangle) \\
&\langle f, g\rangle = (\!|[\langle \underline{4}, \underline{-2}\rangle, \langle (2+) \cdot \pi_2, (2-) \cdot \pi_1\rangle]|\!) \\
&\equiv && (28\text{: Lei da troca}) \\
&\langle f, g\rangle = (\!|\langle [\underline{4}, (2+) \cdot \pi_2], [\underline{-2}, (2-) \cdot \pi_1]\rangle|\!) \\
&\equiv && (53\text{: Fokkinga, F } \langle f, g\rangle = id + \langle f, g\rangle) \\
&\begin{cases} f \cdot \text{in} = [\underline{4}, (2+) \cdot \pi_2] \cdot (id + \langle f, g\rangle) \\ g \cdot \text{in} = [\underline{-2}, (2-) \cdot \pi_1] \cdot (id + \langle f, g\rangle) \end{cases} \\
&\equiv && (\text{in}_{\mathbb{N}_0} = [\underline{0}, \text{succ}], 20\text{: Fusão-+ } (2\times), 22\text{: Absorção-+ } (2\times)) \\
&\begin{cases} [f \cdot \underline{0}, f \cdot \text{succ}] = [\underline{4} \cdot id, (2+) \cdot \pi_2 \cdot \langle f, g\rangle] \\ [g \cdot \underline{0}, g \cdot \text{succ}] = [\underline{-2} \cdot id, (2-) \cdot \pi_1 \cdot \langle f, g\rangle] \end{cases} \\
&\equiv && (1\text{: Natural-id } (2\times), 7\text{: Cancelamento-}\times\ (2\times), 27\text{: Eq-+ } (2\times)) \\
&\begin{cases} \begin{cases} f \cdot \underline{0} = \underline{4} \\ f \cdot \text{succ} = (2+) \cdot g \end{cases} \\ \begin{cases} g \cdot \underline{0} = \underline{-2} \\ g \cdot \text{succ} = (2-) \cdot f \end{cases} \end{cases} \\
&\equiv && (72\text{: Ig. Ext., 73: Def-comp, Def. succ, 75: Def. const}) \\
&\begin{cases} \begin{cases} f\ 0 = 4 \\ f\ (n+1) = 2 + g\ n \end{cases} \\ \begin{cases} g\ 0 = -2 \\ g\ (n+1) = 2 - f\ n \end{cases} \end{cases}
\end{aligned}
$$

Exercício 6

Mostre que a lei da recursividade mútua generaliza a mais do que duas funções, neste caso três:

$$\begin{cases} f \cdot \mathrm{in} = h \cdot \mathrm{F}\ \langle \langle f, g \rangle, j \rangle \\ g \cdot \mathrm{in} = k \cdot \mathrm{F}\ \langle \langle f, g \rangle, j \rangle \\ j \cdot \mathrm{in} = l \cdot \mathrm{F}\ \langle \langle f, g \rangle, j \rangle \end{cases} \quad \equiv \quad \langle \langle f, g \rangle, j \rangle = (\!| \langle \langle h, k \rangle, l \rangle |\!) \tag{F6}$$

$$\begin{cases} f \cdot \mathrm{in} = h \cdot \mathrm{F}\ \langle \langle f, g \rangle, j \rangle \\ g \cdot \mathrm{in} = k \cdot \mathrm{F}\ \langle \langle f, g \rangle, j \rangle \\ j \cdot \mathrm{in} = l \cdot \mathrm{F}\ \langle \langle f, g \rangle, j \rangle \end{cases}$$

$\equiv$ (53: Fokkinga)

$$\begin{cases} \langle f, g \rangle = (\!| \langle h, k \rangle |\!) \\ j \cdot \mathrm{in} = l \cdot \mathrm{F}\ \langle \langle f, g \rangle, j \rangle \end{cases}$$

$\equiv$ $(46\text{: Universal-cata})$

$$\begin{cases} \langle f, g \rangle \cdot \mathrm{in} = \langle h, k \rangle \cdot \mathrm{F}\ \langle \langle f, g \rangle, j \rangle \\ j \cdot \mathrm{in} = l \cdot \mathrm{F}\ \langle \langle f, g \rangle, j \rangle \end{cases}$$

$\equiv$ (53: Fokkinga)

$\langle \langle f, g \rangle, j \rangle = (\!| \langle \langle h, k \rangle, l \rangle |\!)$ c.q.m.

Exercício 7

Considere o functor

$$\begin{cases} \mathrm{T}\ X = X \times X \\ \mathrm{T}\ f = f \times f \end{cases} \tag{F7}$$

e as funções

$$\begin{aligned} \mu &= \pi_1 \times \pi_2 \\ u &= \langle id, id \rangle . \end{aligned} \tag{F8}$$

a) Mostre que T é de facto um functor:

$$\begin{aligned} &\mathrm{T}\ id = id \\ &\mathrm{T}\ (f \cdot g) = \mathrm{T}\ f \cdot \mathrm{T}\ g \end{aligned}$$

b) Demonstre a propriedade:

$$\mu \cdot \mathrm{T}\ u = id = \mu \cdot u$$

Resolução 7

a)

$$\begin{aligned} &\mathrm{T}\ id = id \\ &\equiv && (\mathrm{F7}) \\ &\mathrm{T}\ id = id \times id \\ &\equiv && (15\text{: Functor-id-}\times) \\ &\mathrm{True} \end{aligned}$$

$$\begin{aligned} &\mathrm{T}\ (f \cdot g) = \mathrm{T}\ f \cdot \mathrm{T}\ g \\ &\equiv && (\mathrm{F7}\ (3\times)) \\ &(f \cdot g) \times (f \cdot g) = (f \times f) \cdot (g \times g) \\ &\equiv && (14\text{: Functor-}\times) \\ &\mathrm{True} \end{aligned}$$

b)

$$
\begin{aligned}
&\mu \cdot \mathrm{T}\ u = id \\
&\equiv && (\mathrm{F7}, \mathrm{F8}) \\
&(\pi_1 \times \pi_2) \cdot (\langle id, id \rangle \times \langle id, id \rangle) = id \\
&\equiv && (14\text{: Functor-}\times,\ 7\text{: Cancelamento-}\times\ (2\times)) \\
&id \times id = id \\
&\equiv && (15\text{: Functor-id-}\times) \\
&\mathrm{True}
\end{aligned}
$$

$$
\begin{aligned}
&\mu \cdot u = id \\
&\equiv && (\mathrm{F8}) \\
&(\pi_1 \times \pi_2) \cdot \langle id, id \rangle = id \\
&\equiv && (11\text{: Absorção-}\times,\ 1\text{: Natural-id } (2\times)) \\
&\langle \pi_1, \pi_2 \rangle = id \\
&\equiv && (8\text{: Reflexão-}\times) \\
&\mathrm{True}
\end{aligned}
$$

Exercício 8

Problem requirements:

In the context of a sporting competition (e.g. football league), suppose you have access to the history of all games of the competition, organized by date, in `db1 :: [(Date, [Game])]` *(using Haskell syntax). Also given is* `db2 :: [(Game, [Player])]` *indicating which players played in which game.*

A sport-tv commentator asks you to derive from `db1` *and from* `db2` *the list, ordered by player name, of the dates on which each player played, also ordered. Define, in Haskell, a function* `f` *implementing such a derivation:*

```haskell
f :: [(Date, [Game])] -> [(Game, [Player])] -> [(Player, [Date])]
```

Challenged by these requirements, ChatGPT gave the solution given below in the black text boxes, which doesn't type but is the sort of solution to be expected. In the context of this course, you can write ***far less*** *code to implement* `f` *!*

Why and how?

```haskell
import Data.List (sort, nub)

type Date = String -- You can replace String with an appropriate Date type
type Player = String
type Game = String

-- Helper function to extract unique player names from a list of games
extractPlayers :: [(Game, [Player])] -> [Player]
extractPlayers = nub . concatMap snd

-- Helper function to map players to the dates they played on
mapPlayersToDates :: [(Date, [Game])] -> [(Game, [Player])] -> [(Player, [Date])]
mapPlayersToDates db1 db2 = [(player, sort $ nub playedDates)]
  where
    players = extractPlayers db2
    playedDates player = [date | (date, games) <- db1,
                                 any (\(game, players) -> player `elem` players && game `e

-- Main function f
f :: [(Date, [Game])] -> [(Game, [Player])] -> [(Player, [Date])]
f db1 db2 = mapPlayersToDates db1 db2
```

```haskell
-- Example usage:
main :: IO ()
main = do
  let db1 = [("2023-10-01", ["Game1","Game2"]),
             ("2023-10-02", ["Game2","Game3"])]

  let db2 = [("Game1", ["PlayerA","PlayerB"]),
             ("Game2", ["PlayerA","PlayerC"]),
             ("Game3", ["PlayerB","PlayerC"])]

  let result = f db1 db2
  print result
```