Ejercicios FP

Lucía Napoli 101562

Ejercicio 14

Dada una secuencia de pares ordenados donde la primera componente indica el equipo que resultó ganador y la segunda indica el perdedor y donde cada par ordenado indica un partido jugado (no hay empates) obtener:

a) Los equipos invictos

Lista de partidos: < <RI BO> <RI VE> <IN BO> <IN SL> <BO SL> >

La función me tiene que devolver todos los equipos que están en la primer posición y no en la segunda.

```
Funciones usadas:

Def pertenece = /or o @eq o distl

Def devolver_si_no_pertenece = pertenece -> ~<>; 1

Def insertar_no_vacio = null o 1 -> 2; apndl

Def agregar_vacio = apndr o [id,~<>]

Def insertar_no_vacio = null o 1 -> 2; apndl

Def eliminar_vacio = /insertar_no_vacio o agregar_vacio

Def agregar = pertenece -> 2; apndl

Def er = /agregar o agregar_vacio
```

```
FP> trans: < <RI BO> <RI VE> <IN BO> <IN SL> <BO SL> >
RESULT: <<RI, RI, IN, IN, BO>, <BO, VE, BO, SL, SL>>
           los que ganaron      los que perdieron

FP> distr: <<RI, RI, IN, IN, BO>, <BO, VE, BO, SL, SL>>
RESULT: <<RI, <BO, VE, BO, SL, SL>>, <RI, <BO, VE, BO, SL, SL>>, <IN, <BO, VE, BO, SL, SL>>, <IN, <BO, VE, BO, SL, SL>>,
        <BO, <BO, VE, BO, SL, SL>>>

FP> @devolver_si_no_pertenece: <<RI, <BO, VE, BO, SL, SL>>, <RI, <BO, VE, BO, SL, SL>>, <IN, <BO, VE, BO, SL, SL>>,
    <IN, <BO, VE, BO, SL, SL>>, <BO, <BO, VE, BO, SL, SL>>>
RESULT: <RI, RI, IN, IN, <>>

FP> eliminar_vacio: <RI, RI, IN, IN, <>>
RESULT: <RI, RI, IN, IN>

FP> er: <RI, RI, IN, IN>
RESULT: <RI, IN>

FP> er o eliminar_vacio o @devolver_si_no_pertenece o distr o trans: < <RI BO> <RI VE> <IN BO> <IN SL> <BO SL> >
RESULT: <RI, IN>
```

Entonces defino la función `devolver_invictos`:

```
FP> Def devolver_invictos = er o eliminar_vacio o @devolver_si_no_pertenece o distr o trans
RESULT: FUNCTION DEVOLVER_INVICTOS DEFINED

FP> devolver_invictos: < <RI BO> <RI VE> <IN BO> <IN SL> <BO SL> >
RESULT: <RI, IN>
```

Ejercicio 10

Definir una función selector por izquierda para arreglos de n dimensiones.

Ej: < <3, 2>, < <A, B, C>, <D, E, F>, <G, H, I> > > → <H>

Pasos antes de llegar a la función final:

```
Primero concateno en la primera sec que son los indices, le concateno a la derecha la secuencia con las letras
FP> apndr: < <3, 2>, < <A, B, C>, <D, E, F>, <G, H, I> > >
RESULT: <3, 2, <<A, B, C>, <D, E, F>, <G, H, I>>>

FP> [iota o 2, iota o 1, 3]: <3, 2, <<A, B, C>, <D, E, F>, <G, H, I>>>
RESULT: <<1, 2>, <1, 2, 3>, <<A, B, C>, <D, E, F>, <G, H, I>>>

Pruebas para hacer el selector:
FP> tl: <<1, 2>, <1, 2, 3>, <<A, B, C>, <D, E, F>, <G, H, I>>>
RESULT: <<1, 2, 3>, <<A, B, C>, <D, E, F>, <G, H, I>>>

FP> trans: <<1, 2, 3>, <<A, B, C>, <D, E, F>, <G, H, I>>>
RESULT: <<1, <A, B, C>>, <2, <D, E, F>>, <3, <G, H, I>>>
FP> 1r: <<1, <A, B, C>>, <2, <D, E, F>>, <3, <G, H, I>>>
RESULT: <3, <G, H, I>>
FP> 2: <3, <G, H, I>>
RESULT: <G, H, I>

FP> trans: <<1, 2>, <G, H, I>>
RESULT: <<1, G>, <2, H>>
FP> 1r: <<1, G>, <2, H>>
RESULT: <2, H>
FP> 2: <2, H>
RESULT: H

FP> Def selector = 2 o 1r o trans
RESULT: FUNCTION SELECTOR DEFINED
FP> [1, selector o tl]: <<1, 2>, <1, 2, 3>, <<A, B, C>, <D, E, F>, <G, H, I>>>
RESULT: <<1, 2>, <G, H, I>>
FP> selector: <<1, 2>, <G, H, I>>
RESULT: H

FP> selector o [1, selector o tl]: <<1, 2>, <1, 2, 3>, <<A, B, C>, <D, E, F>, <G, H, I>>>
RESULT: H
```

Funciones usadas:

```
Def iota = auxiota o [~1, id, ~<>]

Def auxiota = (> o [1, 2] -> 3; auxiota o [+ o [1, ~1], 2, apndr o [3, 1]])

Def selector = 2 o 1r o trans

Def sec_indices = [iota o 2, iota o 1, 3] o apndr

Def selector_aux = selector o [1, selector o tl]

Def selector_izq = selector_aux o sec_indices
```

Resultado:

```
FP> selector_izq: < <3, 2>, < <A, B, C>, <D, E, F>, <G, H, I> > >
RESULT: H

FP> selector_izq: < <2, 1>, < <A, B, C>, <D, E, F>, <G, H, I> > >
RESULT: D

FP> selector_izq: < <3, 3>, < <A, B, C>, <D, E, F>, <G, H, I> > >
RESULT: I
```