PLP - Recuperatorio del Primer Parcial - 2do cuatrimestre de 2022

Este examen se aprueba obteniendo al menos dos ejercicios bien menos (B-) y uno regular (R). Las notas para cada ejercicio son: -, I, R, B-, B. Poner nombre, apellido y número de orden en todas las hojas, y numerarlas. Se puede utilizar todo lo definido en las prácticas y todo lo que se dio en clase, colocando referencias claras.
El orden de los ejercicios es arbitrario. Recomendamos leer el parcial completo antes de empezar a resolverlo.

Ejercicio 1 - Programación funcional
Durante este ejercicio **no** se puede usar recursión explícita, a menos que se indique lo contrario. Para resolver un ítem pueden utilizarse las funciones definidas en los ítems anteriores, más allá de si fueron resueltos correctamente o no. Dar el tipo de todas las funciones pedidas.

En este ejercicio trabajaremos con matrices infinitas. Se define el tipo `Matriz` de la siguiente manera:
`data Matriz a = NuevaMatriz a | Agregar a Int Int (Matriz a)`
Donde `NuevaMatriz v` representa una matriz con valor por defecto `v`, y `Agregar v x y m` representa la matriz que resulta de agregar el valor `v` en la fila `x` y columna `y` a la matriz `m`. El valor de una posición de la matriz será el último que se haya agregado en la fila y columna correspondientes, o el valor por defecto si nunca se agregó un valor en esa posición.
Utilizaremos las siguentes matrices para los ejemplos:

```
m1 = Agregar 5 1 2 $ Agregar 2 2 1 $ Agregar 3 1 1 $ NuevaMatriz 0
m2 = Agregar 'a' 1 2 $ Agregar 'b' 1 2 $ Agregar 'c' 1 1 $ NuevaMatriz 'd'
m3 = Agregar 0 1 2 $ Agregar 4 0 0 $ Agregar 2 1 1 $ NuevaMatriz 1
```

a) Definir el esquema de recursión estructural `foldMatriz` para este tipo de matrices, y dar su tipo. En este punto se permite usar recursión explícita.

b) Definir la función `ver::Int->Int->Matriz a->a`, que devuelva el valor de la posición de una matriz correspondiente a la fila y columna indicadas.

Por ejemplo: `ver 1 1 m2` devuelve `'c'`. `ver 1 2 m2` devuelve `'a'`. `ver 0 0 m2` devuelve `'d'`.

c) Definir `suma::Num a =>Matriz a->Matriz a->Matriz a`, que calcule la suma de dos matrices.

Por ejemplo: `suma m1 m3` es la matriz con 4 en la posición (0,0), 5 en las posiciones (1,1) y (1,2), 3 en (2,1) y 1 en todas las demás (porque $0 + 1 = 1$).

Se recomienda definir `suma` por recursión (`foldMatriz`) en la primera matriz que toma como parámetro. Pensar bien cuál es el tipo que va a devolver el fold, y cuál es el resultado de la recursión.

Sugerencia: definir primero el **map** para matrices.

d) Definir la matriz `matrizDeSumas`, que contenga en cada posición con índices naturales la suma de los índices de su fila y columna. El valor por defecto no se usa, por lo que puede ser 0 o un error. Puede colgarse si se intenta ver un valor con índices negativos.

Por ejemplo: `ver 2 3 matrizDeSumas` devuelve 5.

Sugerencia: usar una lista de pares como estructura intermedia.

Ejercicio 2 - Inferencia de tipos Considerar el Cálculo Lambda tipado extendido con listas. Se extiende el cálculo para poder construir listas a partir de un primer elemento, una función generadora y una condición de corte.
El conjunto de tipos no se modifica. El conjunto de términos se extiende de la siguiente manera:

$$M ::= \ldots \mid \mathsf{from}\ M\ \mathsf{until}_x\ M\ \mathsf{by}\ M$$

Una expresión de la forma $\mathsf{from}\ M_1\ \mathsf{until}_x\ M_2\ \mathsf{by}\ M_3$, donde M_1 es un elemento cualquiera, M_2 es una expresión booleana que puede tener libre la variable x, y M_3 una función que se aplicará a cada elemento