PLP - Recuperatorio del Primer Parcial - 1^{er} cuatrimestre de 2016

Este examen se aprueba obteniendo al menos **65 puntos** en total, y al menos **5 puntos** por cada tema. Poner nombre, apellido y número de orden en cada hoja, y numerarlas. Se puede utilizar todo lo definido en las prácticas y todo lo que se dio en clase, colocando referencias claras.

Ejercicio 1 - Programación Funcional (35 puntos)
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'
```

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 la función `mapMatriz::(a->b)->Matriz a->Matriz b` que aplique una función a cada valor de la matriz. Por ejemplo, `mapMatriz (+2) m1` es la matriz con 7 en la posición (1,2), 4 en (2,1), 5 en (1,1) y 2 en todas las demás.

d) Definir la matriz `matrizDePosiciones`, que contenga en cada posición la tupla correspondiente a su fila y columna (el valor por defecto puede ser cualquier par de enteros, ya que no se usa).

   Por ejemplo: `ver 0 1 matrizDePosiciones` devuelve `(0,1)`.

Ejercicio 2 - Extensión de Cálculo Lambda (40 puntos)
Se desea extender el cálculo lambda tipado para soportar relaciones binarias.
Los conjuntos de tipos y de términos se extienden de la siguiente manera:
$\sigma ::= \ldots \mid \mathsf{RelBin}_{\sigma,\tau}$
$M, X, Y, F, G ::= \ldots \mid \mathsf{Vacía}_{\sigma,\tau} \mid \mathcal{R}(X, Y, M) \mid \mathsf{map}(F, G, M)$
Donde $\mathsf{Vacía}_{\sigma,\tau}$ representa a la relación vacía de elementos de tipo σ en elementos de tipo τ, $\mathcal{R}(X, Y, M)$ debe interpretarse como la extensión de la relación M con el vínculo entre los elementos X e Y y por último $\mathsf{map}(F, G, M)$ es la relación obtenida al aplicar la función F sobre todos los elementos del dominio de M y la función G sobre todos los elementos del codominio de M manteniendo los vínculos originales.
Se pide:

a. Introducir las reglas de tipado para la extensión propuesta.

b. Exhibir una derivación para el siguiente juicio de tipado. De no ser posible, explicar el problema.

$$\emptyset \triangleright \mathsf{map}(\lambda x\!:\!\mathsf{Nat}.\mathsf{isZero}(x),\ \lambda y\!:\!\mathsf{Bool}.\lambda z\!:\!\mathsf{Nat}.z,\ \mathcal{R}(0, \mathsf{True}, \mathsf{Vacía}_{\mathsf{Nat},\mathsf{Bool}})) : \mathsf{RelBin}_{\mathsf{Bool},\mathsf{Nat}\to\mathsf{Nat}}$$

c. Indicar formalmente cómo se modifica el conjunto de valores, y dar la semántica operacional de a un paso para la extensión propuesta. Las funciones F y G en las expresiones de la forma $\mathsf{map}(F, G, M)$ no necesitan reducirse hasta ser aplicadas.

d. Mostrar cómo reduce paso por paso el término del inciso (b).

Ejercicio 3 - Inferencia de tipos (25 puntos)

En este ejercicio extenderemos el cálculo-λ^{bn} con la operación de minimización acotada. Para esto, introduciremos términos de la forma $\mathsf{Min}\ x \in [N..M]\ /\ P$, donde x es una variable que puede aparecer libre en P, P es un predicado y N y M delimitan el rango de la minimización. Semánticamente, $\mathsf{Min}\ x \in [N..M]\ /\ P$ representa el mínimo valor de x entre N y M que hace verdadero a P, si existe, o $\mathsf{Succ}(M)$ en caso contrario.

No se modifica el conjunto de tipos, pero se agrega la siguiente regla de tipado:

$$\frac{\Gamma, x : \mathsf{Nat} \triangleright P : \mathsf{Bool} \quad \Gamma \triangleright N : \mathsf{Nat} \quad \Gamma \triangleright M : \mathsf{Nat}}{\Gamma \triangleright \mathsf{Min}\ x \in [N..M]\ /\ P : \mathsf{Nat}}$$

a. Extender el algoritmo de inferencia para soportar la extensión propuesta.

b. Aplicar el algoritmo extendido para tipar la siguiente expresión, o demostrar que no tipa, <u>explicitando las sustituciones realizadas en cada paso</u>:

$$\lambda x.\mathsf{Min}\ t \in [(\mathsf{if}\ x\ \mathsf{then}\ 0\ \mathsf{else}\ \mathsf{Succ}(0))..\mathsf{Succ}(\mathsf{Succ}(0))]\ /\ \mathsf{isZero}(t)$$