PLP - Primer Parcial - 1er cuatrimestre de 2014

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.
Aclaración: No está permitido utilizar recursión explicita en los ejercicios excepto que se indique lo contrario

Ejercicio 1 - Programación funcional (40 puntos)
Una permutación sobre un tipo `a` es una función `f :: a → a` biyectiva:

```
type Perm a = a → a
```

a) Definir la función `permutacionCiclica :: Eq a ⇒[a] → Perm a`
que dada una lista sin repetidos $[x_0, x_1, \ldots, x_{n-1}]$ denota la permutación p tal que

$$\begin{array}{lcll} p(x_i) & = & x_{i+1} & \text{si } 0 \leq i < n-1 \\ p(x_{n-1}) & = & x_0 & \\ p(y) & = & y & \text{si } y \text{ no aparece en la lista} \end{array}$$

Ejemplo: `map (permutacionCiclica [1, 2, 3]) [1, 2, 3, 4]` ⇝ `[2, 3, 1, 4]`

Asumir dada la función `elemIndex :: Eq a ⇒a → [a] → Maybe Int` que devuelve `(Just i)` si `i` es el primer índice del elemento en la lista, y `Nothing` si el elemento no está en la lista.

b) Definir la función `ciclo :: Eq a ⇒Perm a → a → [a]`
que dadas una permutación p y un elemento x_0 en el dominio de p, denota la lista

$$[x_0,\ p(x_0),\ p(p(x_0)), \ldots,\ p^{n-1}(x_0)]$$

donde $n \geq 1$ es el número más chico que cumple $p^n(x_0) = x_0$ (asumiendo que existe tal número).

Ejemplo: `ciclo (permutacionCiclica [1, 2, 3]) 3` ⇝ `[3, 1, 2]`
`ciclo (permutacionCiclica [1, 2, 3]) 4` ⇝ `[4]`

Sugerencia: definir una función auxiliar para calcular p^n y buscar el mínimo n tal que $p^n(x_0) = x_0$.

Las listas de la forma `(ciclo p x0)` se llaman los ciclos de la permutación `p`.

c) Definir la función `rotaciones :: Eq a ⇒[a] → [[a]]`
que dada una lista `xs` finita devuelve todas sus rotaciones. Una lista `ys :: [a]` es una rotación de `xs` sii existen listas `as, bs :: [a]` tales que:
`xs == as ++bs` `ys == bs ++as`

Ejemplo: `rotaciones [1, 2, 3, 4]` ⇝ `[[1, 2, 3, 4], [2, 3, 4, 1], [3, 4, 1, 2], [4, 1, 2, 3]]`

d) Definir la función `todosLosCiclos :: Eq a ⇒[a] → Perm a → [[a]]`
que dado un dominio finito `dom` y una permutación `p`, devuelve una lista con el ciclo de `p` asociado a cada elemento del dominio.

Ejemplo: `todosLosCiclos [1, 2, 3, 4] (permutacionCiclica [1, 2, 3])` ⇝
`[[1, 2, 3], [2, 3, 1], [3, 1, 2], [4]]`

e) Definir la función `descomposicionEnCiclos :: Eq a ⇒[a] → Perm a → [[a]]`
que hace lo mismo que `todosLosCiclos`, pero elimina del resultado final los ciclos repetidos. Un ciclo se considera repetido si él, o cualquiera de sus rotaciones, aparece más de una vez en la lista.

Ejemplo: `descomposicionEnCiclos [1, 2, 3, 4] (permutacionCiclica [1, 2, 3])` ⇝
`[[1, 2, 3], [4]]`

Ejercicio 2 - Cálculo Lambda Tipado (30 puntos)

Se desea extender el cálculo lambda tipado (C-λ^{bn}) con la posibilidad de generar listas infinitas aplicando una función a todos los naturales mayores que un número, y con una construcción para observarlas. Formalmente, se extienden los conjuntos de tipos y términos:

$$\sigma ::= \ldots \mid [\![\sigma]\!]$$

$$M ::= \ldots \mid \mathsf{map}(M,\, [N..]) \mid \mathsf{pop}\ x : \sigma \to M$$

La semántica esperada es la siguiente: $[\![\sigma]\!]$ es el tipo de las listas infinitas de elementos de tipo σ. Si F es una función y N es un número natural, $\mathsf{map}(F,\, [N..])$ denota la lista infinita $[F(N), F(N+1), F(N+2), \ldots]$. Por último, $(\mathsf{pop}\ x : \sigma \to F)$ es una función que espera una lista infinita, liga x a su cabeza, y aplica F a su cola. Notar que F puede contener ocurrencias de x. Por ejemplo, si $I := (\lambda x : \mathsf{Nat}.\ x)$ es la identidad de los naturales:

$$\begin{array}{ll} & (\mathsf{pop}\ x : \mathsf{Nat} \to \lambda y : [\![\mathsf{Nat}]\!].\ x)\, \mathsf{map}(I,\, [\underline{5}..]) \\ \rightsquigarrow & (\lambda y : [\![\mathsf{Nat}]\!].\ I\, \underline{5})\, \mathsf{map}(I,\, [\underline{6}..]) \\ \rightsquigarrow & I\, \underline{5} \quad \rightsquigarrow \quad \underline{5} \end{array}$$

a) Dar las reglas de tipado del cálculo extendido.

b) Indicar cómo se modifica el conjunto de valores e introducir las reglas de semántica operacional para esta extensión.

c) Mostrar la validez del siguiente juicio de tipado utilizando el árbol de derivación:

$$\{f : \mathsf{Nat} \to \sigma\} \ \triangleright\ (\mathsf{pop}\ x : \sigma \to (\lambda x : [\![\sigma]\!].\ x))\, \mathsf{map}(f,\, [\underline{0}..]) : [\![\sigma]\!]$$

Ejercicio 3 - Inferencia de tipos (30 puntos)

A continuación extenderemos el cálculo-λ^{bn} con conjuntos de elementos de la siguiente manera:

$$\sigma ::= ... \mid \mathcal{P}(\sigma)$$

$$M ::= ... \mid \{M_1, \ldots, M_n\} \mid \forall x \in M.\ N$$

La expresión $\{M_1, \ldots, M_n\}$ denota un conjunto con los $n \geq 1$ elementos dados. Por otra parte, si M es un conjunto, la expresión $\forall x \in M.\ N$ denota un booleano que indica si todos los elementos de M cumplen con la condición indicada.

Se extienden las reglas de tipado de la siguiente manera:

$$\frac{\Gamma \triangleright M_i : \sigma \quad \text{para cada } 1 \leq i \leq n}{\Gamma \triangleright \{M_1, \ldots, M_n\} : \mathcal{P}(\sigma)} \qquad \frac{\Gamma \triangleright M : \mathcal{P}(\sigma) \quad \Gamma, x : \sigma \triangleright N : \mathsf{Bool}}{\Gamma \triangleright \forall x \in M.\ N : \mathsf{Bool}}$$

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

b) Utilizar el algoritmo extendido para tipar la siguiente expresión. Recordar que la entrada no contiene anotaciones de tipos.

$$\lambda y.\ \forall x \in y.\ x\, \underline{5}$$