E.T.S.I. Informática. Universidad de Málaga
Examen Ordinario, 26 de Enero de 2006
Programación Declarativa, Prog. Funcional (3º)

APELLIDOS: ____________________

NOMBRE: ____________________

PUNTOS	1-1	1-2	1-3	1-4	2-1	2-2	2-3	3-1	3-2
	3.0	2.0	3.0	2.0	2.0	5.0	3.0	7.0	3.0

☐ si
☐ no
} publique mi calificación si fuera negativa

Consideremos la siguiente estructura para representar los números enteros

data $E \ = \ O \mid S\ E \mid M\ E$ **deriving** *Show*

Por ejemplo, -1 se puede representar en la forma $(M(SO))$ o también en la forma $S(M(S(SO)))$. Sea además la siguiente función de plegado:

$$
\begin{aligned}
&pliega :: (a \to a) \to (a \to a) \to a \to E \to a\\
&pliega\ s\ m\ z\ O \quad\quad\ = z\\
&pliega\ s\ m\ z\ (S\ e) \ = s\ (pliega\ s\ m\ z\ e)\\
&pliega\ s\ m\ z\ (M\ e) = m\ (pliega\ s\ m\ z\ e)
\end{aligned}
$$

1-1 Define la función *par* (que comprueba si un entero es par) directamente a través de *pliega*:

$par :: E \to Bool$
$par = pliega$ ...

1-2 Sea ahora la siguiente definición para la suma de enteros:

instance *Num E* **where**
$$
\begin{aligned}
x + O &= x\\
x + (S\ y) &= S\ (x+y)\\
x + (M\ y) &= M(M\ x + y)
\end{aligned}
$$

Prueba que la igualdad $x + MO = x$ no es demostrable.

1-3 Un dato entero (de tipo E) está normalizado si es, o bien O, o bien de la forma $S^nO (n > 0)$, o bien de la forma $M(S^nO)(n > 0)$. Completa la siguiente función que permite normalizar un entero:

$normaliza :: E \to E$
$normaliza\ O = O$
$normaliza\ (S\ x)$ = **case** $(normaliza\ x)$ **of** $M(S\ O) \to O$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\ \ M(S\ y) \to M\ y$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\ \ $...

$normaliza\ (M\ x)$ = **case** $(normaliza\ x)$ **of** ...

1-4 Define una instancia de la igualdad que permita demostrar por inducción $x + MO == x$:

instance *Eq E* **where** $x == y =$ ...

Consideremos las siguientes definiciones

$$\begin{array}{lcl} bmap\ f\ g\ [\,] & = & [\,] \\ bmap\ f\ g\ (x:xs) & = & f\ x\ :\ g\ x\ :\ bmap\ f\ g\ xs \end{array} \qquad \begin{array}{lcl} length\ [\,] & = & 0 \\ length\ (x:xs) & = & 1\ +\ length\ xs \end{array}$$

2-1 Infiere el tipo más general de la función *bmap*

2-2 Demuestra por inducción sobre listas que *bmap* duplica la longitud de una lista:

$\forall\, xs\ .\ xs :: [a]\ .\ length\ (bmap\ f\ g\ xs)\ =\ 2 * length\ xs\ .$

2-3 Describe la función bmap utilizando solamente la función estándar de plegado de listas *foldr*:

$bmap'\ f\ g\ =\ foldr\ ...$

3-1 Un número binario podemos representarlo como una lista no vacía de elementos del conjunto de caracteres $\{'0',\ '1'\}$. Describe una red de procesos para generar la lista infinita de binarios ordenada por número de bits y por valor:

["0", "1","00", "01", "10", "11", "000", "001", "010", "011", "100", "101", "110", "111", ...

(Ayuda: Usa la función *bmap* del apartado 2-1).

3-2 Completa la siguiente función para el cálculo del mínimo común múltiplo de los elementos de una lista de naturales positivos:

$mcm :: [Integer] \rightarrow Integer$

$mcm\ (x:xs)\ =\ head\ [\ m \mid m\ \leftarrow\ [x,\ 2*x\ ..\ ],$

$\qquad\qquad\qquad\qquad ...$

4-1 Evalúa la expresión: $map\ (take\ 6)\ (map\ mul\ [2,3,4])$, donde $mul\ x\ =\ map\ (*x)\ [1..]$

4-2 Completa las siguientes ecuaciones para una función que calcula los elementos comunes de dos listas estrictamente ascendentes:

$$\begin{array}{l} comunes\ ::\ Ord\ a\ =>\ [a]\ \rightarrow\ [a]\ \rightarrow\ [a] \\ comunes\ (x:xs)\ (y:ys)\ \mid x == y\ =\ x\ :\ comunes\ xs\ ys \\ \qquad\qquad\qquad\qquad\qquad\ \ \mid x < y\ \ \ =\qquad comunes\ xs\ (y:ys) \\ \qquad\qquad\qquad\qquad\qquad\ \ \mid ... \\ ... \end{array}$$

4-3 Demuestra que $comunes\ xs\ ys$ computa en forma estrictamente ascendente los elementos comunes de sus argumentos si éstas son dos listas ordenadas en forma estrictamente ascendente (Ayuda.- Use inducción sobre pares de listas con el orden lexicográfico entre pares.)

4-4 Sean las funciones

$$\begin{array}{l} inter\ [xs]\ =\ xs \\ inter\ (xs : ys : rs)\ =\ comunes\ xs\ (inter\ (ys : rs)) \end{array}$$

$$aj\acute{a}\ =\ head\ .\ inter\ .\ map\ mul$$

¿Cuales son sus tipos?

$$inter\ ::\ ...$$

$$aj\acute{a}\ \ ::\ ...$$

¿Que valores computan las funciones anteriores?