Programación declarativa **(29 de noviembre de 2010)**

Apellidos:

Nombre:

Grupo:

Observaciones:

1. *En la evaluación se tendrá en cuenta la corrección, simplicidad y eficiencia de la respuesta.*
2. *Hay que describir las definiciones auxiliares (menos las del sistema).*
3. *Indicar el tipo más genérico de cada función definida.*

Ejercicio 1 [2 pto] Contestar en los espacios en blanco.

1. ¿Qué es un operador?

2. ¿Cuando se puede convertir una función en un operador? ¿cómo se hace?

3. Calcular el valor de las siguientes expresiones:

```
tail [[1,2],[]] ++ [head [[1,2],[]]]
```

```
([1,2,3,4,5] !!  3):[1,2,3]
```

```
or [(<= 3) x | x <- [2,4..]]
```

```
filter (<= 5) (map (length .  init) ["casa", "avion",
"relojes"])
```

4. Consideremos las siguientes definiciones

```
f1 ::  Int -> [Int]
f1 0 = [0]
f1 (n+1) = (n+1):(f1 n)
f2 ::  (Num b) => [a] -> b -> b
f2 [] ac = ac
f2 (x:xs) ac = f2 xs (1+ac)
```

Demostrar que para cualquier número natural, `n`, se tiene que para todo número `ac`: `f2 (f1 n) ac == 1+ac+n`

Programación declarativa **(29 de noviembre de 2010)**

Apellidos:

Nombre:

Grupo:

Ejercicio 2 [2 puntos] Consideremos el siguiente tipo definido para representar arboles binarios:

```
data Arbol a = Hoja a | Nodo (Arbol a) a (Arbol a)
```

1. Definir una función

   ```
   altura ::  Arbol a -> Int
   ```

   que dado un árbol binario calcule su altura. Ejemplo:

   ```
   altura (Nodo (Nodo (Hoja 1) 3 (Hoja 4)) 5 (Hoja 6))
   ==> 2
   ```

2. Definir una función

   ```
   datos ::  Arbol a -> [a]
   ```

   que dado un árbol binario devuelva la lista de los elementos que aparecen en el mismo leídos de izquierda a derecha y de arriba a abajo.

   ```
   datos (Nodo (Nodo (Hoja 1) 3 (Hoja 4)) 5 (Hoja 6)) ==>
   [5,3,1,4,6]
   ```

Ejercicio 3 [2 puntos] Definir una función `elimina`, que dadas una lista, `ls`, y un número natural, `n`, devuelva la lista que resulta de eliminar de `ls` todos los elementos cuyas posiciones son múltiplos de `n`.
Dar una definición utilizando listas por compresión y otra utilizando recursión.
Ejemplos:

```
elimina [1,2,3,4,5,6,7,8,9] 3 ==> [2,3,5,6,8,9]
elimina "Esta cadena" 5 ==> "sta aden"
```

Programación declarativa **(29 de noviembre de 2010)**

Apellidos:

Nombre:

Grupo:

Ejercicio 4 [2 puntos]

Definir una función `maximo` que dada una función, `f`, y una lista, `ls`, devuelva aquel elemento de `ls` en el que `f` alcanza su valor máximo. Ejemplos:

```
maximo sqrt [3, 1, 6, 2, 8, 9] ==> 9.0
maximo abs [3, 1, (-6), 2, 8, (-9)] ==> -9
```

Ejercicio 5 [2 puntos]

Definir una función `agrupa` que dada una lista (sin elementos repetidos), `l1` y otra lista, `l2`, cuyos elementos aparecen en `l1`, devuelva una lista de pares que asocie a cada elemento de `l1` el número de veces que aparece en `l2`. Ejemplo:

```
agrupa "aser" "asasaas" => [('a',4),('s',3),('e',0),('r',0)]
```