Programación declarativa (11–Febrero–2010)

Apellidos:

Nombre:

Observaciones:

1. *En la evaluación se tendrá en cuenta la corrección, simplicidad y eficiencia de las respuestas.*
2. *Hay que describir las definiciones auxiliares (menos las del sistema).*

Ejercicio 1 (2 puntos) Definir la función

```
pertenece :: Ord a => a -> [a] -> Bool
```

tal que `(pertenece x ys)` se verifica si `x` pertenece a la lista ordenada creciente, finita o infinita, `ys`. Por ejemplo,

```
pertenece 23 [1,3..]     ==> True
pertenece 22 [1,3..]     ==> False
pertenece 22 [1,3,22,34] ==> True
pertenece 22 [1,3,34]    ==> False
```

Solución:

```
pertenece :: Ord a => a -> [a] -> Bool
pertenece _ [] = False
pertenece x (y:ys) | x >  y    = pertenece x ys
                   | x == y    = True
                   | otherwise = False
```

Ejercicio 2 (2 puntos) Definir la constante

```
primos :: [Int]
```

tal que `primos` es la lista de los primos obtenida mediante la criba de Erastótenes. Por ejemplo,

```
take 15 primos ==> [2,3,5,7,11,13,17,19,23,29,31,37,41,43,47]
```

Solución:

```
primos = criba [2..]

criba :: [Int] -> [Int]
criba (p:xs) = p : criba [x | x <- xs, x `mod` p /= 0]
```

Ejercicio 3 (1.5 puntos) Calcular el valor de las siguientes expresiones:

1. `map (^2) [5,3,8]`
2. `foldr (:) [10] [5,3,8]`
3. `take 5 [(x,y) | x <- [1..], y <- [1..x], even (x+y)]`

4. `maximum (concat [[1..x] | x <- [1..100]])`

Solución: Los valores son

1. `[25,9,64]`
2. `[5,3,8,10]`
3. `[(1,1),(2,2),(3,1),(3,3),(4,2)]`
4. `100`

Ejercicio 4 (1.5 puntos) Se considera el programa lógico:

```
p(0,0).                    %R1
p(s(X),s(Y)) :- !, p(X,Y). %R2
p(s(X),Y) :- p(X,Y).       %R3
```

Construye el árbol de resolución para el programa anterior y la pregunta:

```
?- p(s(s(0)),X).
```

Ejercicio 5 (1.5 puntos) Definir la función

```
divisiones :: [a] -> [([a],[a])]
```

tal que `(divisiones xs)` es la lista de las divisiones de `xs` en dos listas no vacías. Por ejemplo,

```
*Main> divisiones "bcd"
[("b","cd"),("bc","d")]
*Main> divisiones "abcd"
[("a","bcd"),("ab","cd"),("abc","d")]
```

Solución:

```
divisiones :: [a] -> [([a],[a])]
divisiones []     = []
divisiones [_]    = []
divisiones (x:xs) = ([x],xs) : [(x:is,ds) | (is,ds) <- divisiones xs]
```

Ejercicio 6 (1.5 puntos) Definir la función

```
esEleccion :: Eq a => [a] -> [a] -> Bool
```

tal que `(esEleccion xs ys)` se verifica si `xs` es una sublista de `xs` en cualquier orden. Por ejemplo,

```
esEleccion "ec"  "bcde"   ==>  True
esEleccion "ece" "bcde"   ==>  False
esEleccion "eca" "bcde"   ==>  False
```

Solución:

```
esEleccion :: Eq a => [a] -> [a] -> Bool
esEleccion [] _      = True
esEleccion (x:xs) ys = elem x ys && esEleccion xs (eliminaUna x ys)
```

donde `(eliminaUna x ys)` es la lista obtenida eliminando la primera ocurrencia de `x` en `ys`

```
eliminaUna :: Eq a => a -> [a] -> [a]
eliminaUna _ []                 = []
eliminaUna x (y:ys) | x == y    = ys
                    | otherwise = y : eliminaUna x ys
```