Programación declarativa (21 de enero de 2008)

Ejercicio 1 Define en Haskell, *usando listas definidas por comprensión*, la función:

```
misterioH ::  (a -> Bool) -> [a] -> Bool
```

que calcula el predicado `misterioP(+P,+L)` definido en Prolog como sigue:

```
misterioP(P,L) :- member(X,L),apply(P,[X]).
```

Solución:

```
misterioH p xs = or [p x | x <- xs]
```

Ejercicio 2 [2 puntos] Define en Haskell la función:

```
inserta ::  a -> [a] -> Int -> [a]
```

tal que `(inserta x xs n)` inserta el elemento `x` en la lista `xs` en la posición `n`. Por ejemplo:

```
inserta 'X' "abcd" 2 ⇝ "aXbcd"
inserta 'X' "abcd" 10 ⇝ "abcdX"
inserta 'X' "abcd" (-4) ⇝ "Xabcd"
```

Escribe una definición *por recursión* y otra usando las funciones del preludio `take` y `drop`.

Solución:

1. Por recursión:

```
inserta1 x xs n | n <= 1 = x:xs
inserta1 x [] n = [x]
inserta1 x (y:xs) n = y :  inserta1 x xs (n-1)
```

2. Usando `take` y `drop`

```
inserta2 x xs n = take (n-1) xs ++ [x] ++ drop (n-1) xs
```

Ejercicio 3 [2 puntos] Se define el tipo de datos `Arbol` como sigue:

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

Define en Haskell la función

```
sumaTotal ::  Num a => Arbol a -> a
```

tal que `(sumaTotal a)` devuelve el resultado de sumar todos los números que aparecen en los nodos o las hojas del árbol `a`. Por ejemplo:

```
sumaTotal (Nodo 5 (Hoja 7) (Nodo 1 (Hoja 2) (Hoja 6))) ⇝ 21
```

Solución:

```
sumaTotal (Hoja x) = x
sumaTotal (Nodo x a1 a2) = x + sumaTotal a1 + sumaTotal a2
```

Ejercicio 4 [1 punto] Define en Prolog un predicado `misterioP(+L,-N)` que calcule la función definida en Haskell como sigue:

```
misterioH ::  [Int] -> Int
misterioH xs = foldl (+) 0 (map (*2) xs)
```

Solución:

```
misterioP([],0).
misterioP([X|L],N) :- misterioP(L,N1),X1 is X*2,N is N1+X1.
```

Ejercicio 5 Se considera el siguiente programa lógico:

```
p([],[]).
p([X|A],[X|B]) :- q(X),!,p(A,B).
p([X|A],B) :- p(A,B).
q(a).
q(c).
```

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

   ```
   ?- p([a,b],L).
   ```

2. Indica razonadamente qué responde Prolog a la siguiente pregunta.

   ```
   ?- findall(X,not(q(X)),L).
   ```

Solución:

1. Véase la última página.
2. Puesto que a la pregunta `?- q(X).` responde `X = a`, a la pregunta `?- not(q(X)).` responde `No`. Por tanto, a la pregunta del enunciado responde `L = []`.

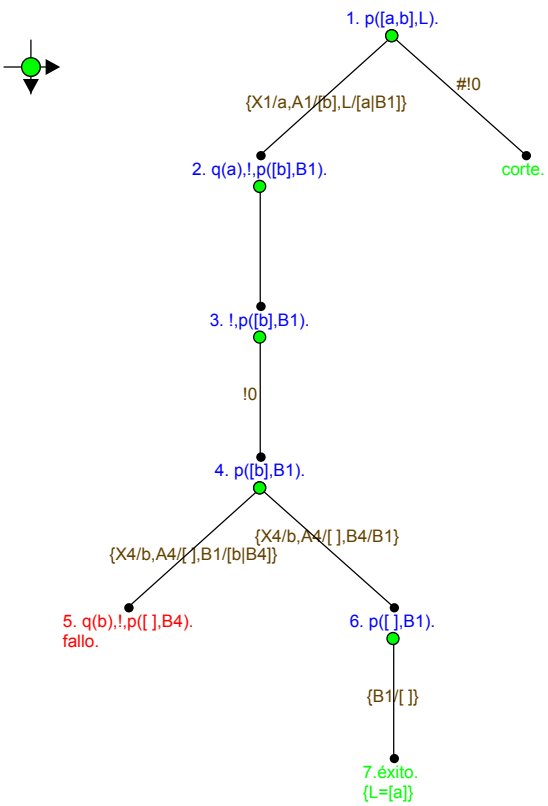
Ejercicio 6 [2 puntos] Define en Prolog el predicado `alarga(+F1,+N,-F2)` que se verifica si F1 y F2 son figuras del mismo tipo y el tamaño de F2 es el de F1 multiplicado por N. Por ejemplo:

```
?- alarga(triangulo(3,4,5),2,F).
F = triangulo(6,8,10)
?- alarga(cuadrado(3),2,F).
F = cuadrado(6)
```

Solución:

```
alarga(F1,N,F2) :- F1 =.. [Tipo|L1], multiplica(L1,N,L2),
                   F2 =.. [Tipo|L2].

multiplica([],N,[]).
multiplica([X1|L1],F,[X2|L2]) :- X2 is X1*F, multiplica(L1,F,L2).
```


1. p([a,b],L).
#!0
{X1/a,A1/[b],L/[a|B1]}
2. q(a),!,p([b],B1).
corte.
3. !,p([b],B1).
!0
4. p([b],B1).
{X4/b,A4/[ ],B4/B1}
{X4/b,A4/[ ],B1/[b|B4]}
5. q(b),!,p([ ],B4).
fallo.
6. p([ ],B1).
{B1/[ ]}
7.éxito.
{L=[a]}