Programación declarativa (21–Diciembre–2000)

Apellidos:

Nombre:

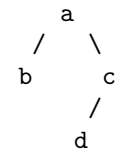
Ejercicio 1 Un árbol binario es vacío o consta de tres partes:

- Una raíz
- Un subárbol izquierdo (que debe ser un árbol binario)
- Un subárbol derecho (que debe ser un árbol binario)

Consideraremos la siguiente representación

- `nil` representa el árbol vacío
- Si el árbol no es vacío, entonces tendrá la forma `t(I,R,D)` donde `R` es la raíz, `I` el subárbol izquierdo y `D` el subárbol derecho.

Por ejemplo, el árbol



lo representamos como `t(t(nil,b,nil),a,t(t(nil,d,nil),c,nil))`. El ejercicio tiene los siguientes apartados:

(a) [0.5 puntos] Definir el predicado `arbolbinario(A)` que se verifique si `A` es un árbol binario. Por ejemplo,

```
?- arbolbinario(nil).
Yes
?- arbolbinario(t(t(nil,b,nil),a,t(t(nil,d,nil),c,nil))).
Yes
?- arbolbinario(t(t(nil,b,nil),a,t(t(nil,d,e),c,nil))).
No
```

(b) [0.5 puntos] Definir el predicado `en(Nodo,Arbol)` que se verifique si `Nodo` es un nodo del árbol binario `Arbol`. Por ejemplo,

```
?- en(Nodo,t(t(nil,b,nil),a,t(t(nil,d,nil),c,nil))).
Nodo = a ;
Nodo = b ;
Nodo = c ;
Nodo = d ;
No
```

(c) [1 punto] Definir el predicado `profundidad(Arbol,Prof)` que se verifique si `Prof` es la profundidad nodo del árbol binario `Arbol`. Consideraremos que la profundidad del árbol vacio es 0 y la del árbol con un único nodo es 1. Por ejemplo,

```
?- prof(t(t(nil,b,nil),a,t(t(nil,d,nil),c,nil)),Prof).
Prof = 3 ;
No
```

(d) [1 punto] Definir el predicado `alinea(Arbol,Lista)` que se verifique si `Lista` es la lista de los nodos del `Arbol`. Por ejemplo,

```
?- alinea(t(t(nil,b,nil),a,t(t(nil,d,nil),c,nil)),L).
L = [b, a, d, c] ;
No
```

Ejercicio 2 [2 puntos] Definir el predicado `sustituye(Sub1,Term1,Sub2,Term2)` que se verifique si `Term2` es el término obtenido sustituyendo todas las ocurrencias de `Sub1` en `Term1` por `Sub2`. Por ejemplo,

```
?- sustituye(sen(x),2*sen(x)*f(sen(x)),y,T).
T = 2 * y * f(y)
Yes

?- sustituye(a+b,f(a,A+B),c,T).
A = a
B = b
T = f(a, c)
Yes
```

Ejercicio 3 [3 puntos] Definir un predicado `ultimos(L,L_ultimos)` que recibe una lista `L` y devuelve la lista `L_ultimos` formada por los últimos elementos de las listas que contenga L, a cualquier nivel. Por ejemplo,

```
?- ultimos([[1,2,3],[4,[5,6]],7],L).
L = [3, [5, 6], 6] ;
No
?- ultimos([1,2,[],3],L).
L = [] ;
No
```

Ejercicio 4 [1 punto] Definir el predicado `producto_cartesiano(+L1,+L2,-Producto)` tal que tome como entrada las listas `L1` y `L2` y devuelva la lista `Producto` formada por todos los pares ordenados del producto cartesiano de `L1` y `L2`. Cada par se representará como una lista de dos elementos. Por ejemplo,

```
?- producto_cartesiano([1,2,3],[a,b],L).
L = [[1, a], [1, b], [2, a], [2, b], [3, a], [3, b]] ;
No
```

Ejercicio 5 [1 puntos] Considera el predicado `misterio(L1, L2)` que toma como entrada la lista `L1` y devuelva la lista `L2` ¿Qué relación hay entre ambas listas?

```
misterio(L1, L2) :-
  aux(L1, L2, []),

aux([], R, R).
aux([X|B], [X|R], T) :-
   aux(B, R, [X|T]).
```