Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
CI3661 - Taller de Lenguajes de Programación I
Enero-Marzo 2015

Quiz Prolog

(10 puntos)

Nombre:

Carnet:

Atención: en todas sus respuestas debe utilizar *exclusivamente* predicados disponibles en el estándar ISO provisto por GNU Prolog. Si Ud. no sabe si un predicado está o no en el conjunto estándar ISO provisto por GNU Prolog, puede preguntarlo *públicamente* para recibir un **si** o **no** como respuesta y **nada** más.

1. **(3 puntos)** Implante el predicado `esrever(Lista,Atsil)` que triunfe *exactamente una vez*, si `Atsil` corresponde a invertir las posiciones de los elementos de `Lista`. Puede suponer que al menos uno de los dos argumentos estará unificado. Por ejemplo,

```
?- esrever([1,2,3],Atsil).
L1Atsil = [3,2,1]
yes
?- esrever(Lista,[3,2,1]).
Lista = [1,2,3]
yes
?- esrever([1,2,3],[3,2,1]).
yes
```

   No puede usar el predicado `append/3` ni directa ni indirectamente.

```
esrever(Lista,Atsil) :- doit(Lista,Atsil,[]).
doit([],R,R) :- !.
doit([H|T],R,A) :- doit(T,R,[H|A]).
```

2. **(3 puntos)** Implante el predicado `divisores(N,Divisores)` que triunfe *exactamente una vez* si `Divisores` corresponde a la lista de divisores de `N`, ordenados de menor a mayor. El predicado debe fallar si `N` está unificado con cualquier cosa que no sea un número entero positivo. Por ejemplo,

```
?- divisores(6,Divisores).
Divisores = [1,2,3,6]
yes
?- divisores(6,[1,2,3,6]).
yes
?- divisores(6,[1,2,3]).
no
?- divisores(0,Divisores).
no
```

Pista: Prolog **tiene** los functores `div/2` y `mod/2`, que pueden usarse en combinación con `is/2` para calcular el cociente y residuo de división entera, respectivamente.

```
divisores(N,Divisores) :- integer(N), N > 0,
                          go(1,N,[],D), esrever(D,Divisores).
go(N,N,A,[N|A]) :- !.
go(M,N,A,D) :- R  is mod(N,M), R = 0, !,
               M1 is M + 1, go(M1,N,[M|A],D).
go(M,N,A,D) :- M1 is M + 1, go(M1,N,A,D).
```

3. **(4 puntos)** Un número entero es perfecto cuando es igual a la suma de sus divisores propios. Implante el predicado `perfecto(N)` que triunfe si `N` es un número perfecto. Si `N` está instanciado con un número entero, el predicado debe triunfar *exactamente una vez* si el número es perfecto y fallar si no lo es. Si `N` es una variable libre, entonces el predicado debe producir los infinitos números perfectos aprovechando *backtracking*.

```
?- perfecto(6).
yes
?- perfecto(7).
no
?- perfecto(N).
N = 6 ? ;
N = 28 ? ;
N = 496 ?

perfecto(N) :- integer(N), !,
               divisores(N,D),
               sum(D,T),
               N =:= T - N.
perfecto(N) :- var(N), !,
               natural(N1),
               perfecto(N1),
               N = N1.
natural(1).
natural(N) :- natural(N1), N is N1 + 1.
sum([],0) :- !.
sum([D|R],S) :- sum(R,T), S is D + T.
```