Programación Declarativa (Lógica y Restricciones)

Fecha de entrega: 10 de marzo

Ejercicios de Programación Lógica Pura
Parte 2 (Peano y listas)

Ejercicio 1 Dada la definición estándar de número natural:

```
natural(0).
natural(s(X)) :-
    natural(X).
```

definir los siguientes predicados (nota: algunos pueden estar ya en las transparencias o en el código que acompaña a las transparencias, pero intentad hacerlos!):

a) `suma(X,Y,Z)`, cierto si y solo si `Z` es la suma de `X` e `Y`.

b) `par(X)`, cierto si y sólo si `X` es par.

c) `impar(X)`, cierto si y sólo si `X` es impar.

Ejercicio 2 Utilizando los predicados del ejercicio anterior, escribir también los siguientes predicados:

a) `suma_a_lista(L,N,SL)`, cierto sí y sólo si `SL` es la lista que se obtiene al sumarle `N` a cada uno de los elementos de la lista `L`.

b) `pares_lista(L,Ps)`, cierto sí y sólo si `Ps` es una lista que contiene los números que son pares de la lista L.

Ejercicio 3 Escribir un predicado puro `extrae_elemento(I,L,E,NL)` que es cierto si `I` es un índice (un natural en notación de Peano), `L` una lista, `E` el elemento de `L` que está en la posición indicada por el índice `I`, y `NL` es lista `L` pero sin ese elemento `E`. El primer elemento de la lista `L` es la posición `0`. Por ejemplo, dado el índice `s(s(0))` y la lista `[a,b,c,d,e]` el elemento `E` es `c` y la lista `NL` es `[a,b,d,e]`.

Importante: Para esta tarea el archivo `code.pl` *debe comenzar con una declaración de módulo* de la forma:

```
:- module(_,_,[pure]).
```

El argumento `[pure]` determina el características del lenguaje que se puede utilizar en el fichero `code.pl`, en particular `pure` evita cargar cualquier predicado built-in de Prolog, de modo que sólo se permite programación lógica pura.