Primera práctica (Programación lógica pura)

Fecha de entrega: **9 de abril** de 2021

SUMAS DE PARES DE LISTAS Y CUADRADOS

Enunciado

Parte 1 (7 puntos): Nuestro primer objetivo es, dado un natural `N` par, construir dos listas de longitud `N/2`, cuyos elementos sumen lo mismo. Para ello, definir los siguientes predicados:

1. Definir un predicado `nums(N,L)` tal que `N` es un número natural y `L` es la lista de números naturales en orden descendente de N a 1.
2. Definir un predicado `sumlist(L,S)` tal que `L` es una lista de números naturales y `S` es la suma de todos los elementos de `L`.
3. Definir un predicado `choose_one(E,L,R)` tal que `L` es una lista, `E` es un elemento cualquiera de `L`, y `R` es lo que queda de la lista, después de quitar `E`. Dada una lista `L`, una llamada a `choose_one/3` debe devolver en `E`, como alternativas al pedir más soluciones, sucesivamente todos los elementos de `E`, y en `R` todos los restos.
4. Usando el predicado `choose_one/3`, escribir el predicado `perm(L,LP)`, tal que `L` es una lista y `LP` es una permutación de `L` (es decir, una lista con los mismos elementos de `L`, en distinto orden). Por ejemplo, dada una lista `L`, `perm(L,LP)` debe generar como alternativas en `LP` todas las permutaciones de `L`.
5. Definir un predicado `split(L,L1,L2)` tal que `L` es una lista de longitud `N`, `N` es par, `L1` contiene los `N/2` elementos en posición impar de `L` y `L2` los en posición par. Es decir: `split([a,b,c,d],X,Y)` devolvería `X = [a,c], Y = [b,d]`.
6. Para completar nuestro primer objetivo, escribir, usando los predicados anteriores, un predicado `sumlists(N,L1,L2,S)` tal que `N` es par, `L1` y `L2` son dos listas de longitud `N/2`, que contienen entre ellas todos los números de Peano de 1 a N, y `L1` y `L2` suman lo mismo. `S` debe ser el valor de dicha suma.

Parte 2 (3 puntos): Como segundo objetivo, dado `N`, y los números de Peano consecutivos de 1 a N^2, colocarlos en un cuadrado de tamaño $N \times N$ tal que todas las filas sumen lo mismo. Para ello programar (pudiéndose usar algunos predicados del punto anterior) el siguiente predicado:

> `square_lists(N,SQ,S)`, tal que `N` es el número, `SQ` el cuadrado (representado como `N` listas de `N` elementos cada una), y `S` el valor que suman las filas.
> Por ejemplo, `square_lists(s(s(0)),SQ,S)` devolvería `S = s(s(s(s(0)))), SQ = [ [s(s(s(0))), s(0)], [s(s(0)), s(s(0))] ]`.

PUNTOS ADICIONALES (subir nota):

- Ejercicio complementario en '*programación alfabetizada*' ('literate programming'):

  Se darán puntos adicionales a las prácticas que realicen la documentación de dichos predicados insertando en el código aserciones y comentarios del lenguaje Ciao, y entregando como memoria o parte de ella el manual generado automáticamente a partir de dicho código, usando la herramienta `lpdoc` del sistema Ciao. Se recomienda intentar escribir este manual de forma que sustituya completamente a la memoria.

- Ejercicio complementario en *codificación de casos de prueba*:

  También se valorará (sólo para subir nota) el uso de aserciones `test` que enumeren casos de prueba para comprobar el funcionamiento de los predicados.

Hemos dejado en Moodle instrucciones específicas sobre cómo hacer todo esto y un ejemplo de código que tiene ya comentarios y tests, para practicar corriendo lpdoc sobre él y ejecutando los tests.

Instrucciones generales para la realización y entrega las prácticas

Es **muy importante** leer el **documento con este título en Moodle**.

Instrucciones específicas

Además de dichas instrucciones generales, como instrucciones específicas para esta práctica se recuerda que debido a que esta práctica corresponde a la parte de programación lógica pura no está permitido el uso de recursos de ISO-Prolog que van más allá, tales como, p.ej., aritmética con `is/2`, predicados metalógicos, negación por fallo, cortes, etc.