PROGRAMACIÓN DECLARATIVA: LÓGICA Y RESTRICCIONES
Grado en Ingeniería Informática / Grado en Matemáticas e Informática

27 de Junio de 2014 **Convocatoria de Julio**

Nombre:

Nº de Matrícula:

INSTRUCCIONES: El examen consta de 3 ejercicios. Todas las preguntas deben comenzar a contestarse en su hoja correspondiente. Pueden añadirse al ejercicio tantas hojas como sean necesarias siempre que estén numeradas y lleven el nombre y número de matrícula del alumno.

DURACIÓN DEL EXAMEN: 90 minutos

EJERCICIO 1 (3'5 puntos – 30 minutos)

La estructura de árbol viene definida por: (a) `empty` para el árbol vacío, (b) `branch`(I,N,D) para un árbol con raíz N, hijo izquierdo I e hijo derecho D.

(A) Programar el predicado **post_orden/2** definido como: post_orden(A,L) si y solo si L es la lista de los elementos del árbol A en el orden en que se visitan en un recorrido en post-orden. (*2 puntos*)

(B) Dar otra versión del mismo predicado **post_orden/2** que no utilice llamadas explícitas a ningún predicado de concatenación. (*2 puntos*)

SOLUCIÓN:

(A)

```
post_orden(empty,[]).
post_orden(branch(I,N,D),L) :-
        post_orden(I,LI),
        post_orden(D,LD),
        append(LD,[X],LL),
        append(LI,LL,L).
```

(B)

```
post_orden(A,L) :- post_orden(A,L,[]).

post_orden(empty,L,L).
post_orden(branch(I,N,D),L,Resto) :-
        post_orden(I,L,LD),
        post_orden(D,LD,[X|Resto]).
```

PROGRAMACIÓN DECLARATIVA: LÓGICA Y RESTRICCIONES
Grado en Ingeniería Informática / Grado en Matemáticas e Informática

27 de Junio de 2014 **Convocatoria de Julio**

Nombre:

Nº de matrícula:

DURACIÓN DEL EXAMEN: 90 minutos

INSTRUCCIONES: El examen consta de 3 ejercicios. Todos los ejercicios deben comenzar a contestarse en su hoja correspondiente. Pueden añadirse al ejercicio tantas hojas como sean necesarias siempre que estén numeradas y lleven el nombre y número de matrícula del alumno.

EJERCICIO 2 (3,5 puntos – 30 minutos)

Dada la siguiente estructura para representar jugadores que pertenecen a equipos de futbol, *miembro_equipo(Jugadores, Equipo)*, donde

- el primer argumento es una estructura que indica el rol de los jugadores con tantos argumentos como jugadores tengan ese rol, y
- el segundo argumento es el equipo al que pertenecen los jugadores, que se representa mediante el functor equipo/1 cuyo argumento es una constante que identifica al equipo.

Se pide definir el predicado **seleccion_jugadores/1** (seleccion_jugadores(-JugadoresSeleccionados)) que lee una lista de equipos favoritos (vía teclado) y un rol favorito (también vía teclado), y que se verifica si JugadoresSeleccionados es la lista de los jugadores con el rol favorito que pertenecen a los equipos favoritos.

Ejemplo: Suponiendo los siguientes hechos:

miembro_equipo(delanteros(ronaldo, benzema), equipo(realmadrid)).

miembro_equipo(porteros(casillas, dlopez), equipo(realmadrid)).

miembro_equipo(defensas(ramos), equipo(realmadrid)).

miembro_equipo(delanteros(messi), equipo(barsa)).

miembro_equipo(porteros(pinto), equipo(barsa)).

miembro_equipo(delanteros(costa, villa), equipo(atleticomadrid)).

miembro_equipo(porteros(courtois), equipo(atleticomadrid)).

?- seleccion_jugadores(JugadoresSeleccionados).

|: [equipo(realmadrid), equipo(atleticomadrid)].

|: delanteros.

JugadoresSeleccionados = [ronaldo,benzema,costa,villa]

SOLUCIÓN:

```
seleccion_jugadores(Jugadores):-
        write('Dime por favor tus equipos favoritos'),
        read(Equipos),
        write('Dime por favor tu rol favorito'),
        read(Rol),
        findall(J,
        (member(equipo(E),Equipos),miembro_equipo(J,equipo(E)),J=..[R|_Js],R==Rol),Jugs),
        recorrer(Jugs,Jugadores).

recorrer([],[]).
recorrer([X],L):-
        X=..[_|Jugs],
        L = Jugs.
recorrer([X|R],L):-
        X=..[_|Jugs],
        recorrer(R,L1),
        append(Jugs,L1,L).
```

PROGRAMACIÓN DECLARATIVA: LÓGICA Y RESTRICCIONES
Grado en Ingeniería Informática / Grado en Matemáticas e Informática

27 de Junio de 2014 **Convocatoria de Julio**

Nombre:

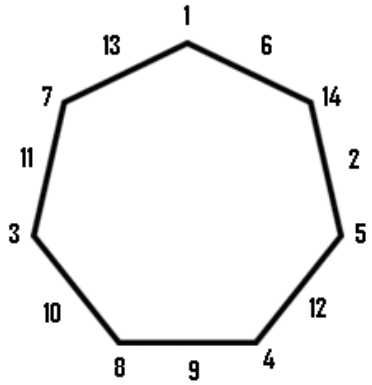
Nº de matrícula:

INSTRUCCIONES: El examen consta de 3 ejercicios. Todas las preguntas deben comenzar a contestarse en su hoja correspondiente. Pueden añadirse al ejercicio tantas hojas como sean necesarias siempre que estén numeradas y lleven el nombre y número de matrícula del alumno.

DURACIÓN DEL EXAMEN: 90 minutos

EJERCICIO 3 (3 puntos – 30 minutos)

Henry Dudeney (1847-1930) propuso numerosos problemas matemáticos durante su vida. Uno de ellos, conocido como el problema del heptágono, entraña cierta complejidad de resolución. Se trata de un heptágono donde cada arista del mismo debe ser etiquetada con números entre el 1 y el 14 de la siguiente manera: un número en cada vértice de la arista y otro en el centro de la arista. El etiquetado debe realizarse de tal forma que el resultado de sumar el número asignado al centro de la arista con los números asignados a los dos vértices extremos sea el mismo para todas las aristas. Nótese que los vértices pertenecientes a dos aristas diferentes solo deben ser etiquetados una vez. Además, un número no puede asignarse más de una vez (ya sea a un vértice o a un centro de arista), y todos los números entre el 1 y el 14 deben ser asignados a un vértice o centro de arista. La figura que sigue a continuación muestra una posible solución al problema, donde la suma mencionada con anterioridad es 21.



Se pide programar un predicado con restricciones en dominios finitos —ya sea utilizando la sintaxis de Ciao Prolog o de SWI-Prolog— ***resolver_heptagono/2*** que recibe como primer argumento una lista de 14 variables donde se almacenarán los números asignados a cada vértice y arista del heptágono una vez resuelto, y como segundo argumento una variable que denota la suma asociada a cada arista —y que debe ser igual para todas las aristas del heptágono etiquetado. Por simplicidad, se recomienda al alumno que utilice la lista *[V1, A12, V2, A23, V3, A34, V4, A45, V5, A56, V6, A67, V7, A71]* como primer argumento del predicado *resolver_heptagono/2*, donde Vi representa al vértice i-ésimo y Aij representa a la arista que está delimitada por los vértices i-ésimo y j-ésimo.

Ejemplo (el de la figura):

?- resolver_heptagono([V1, A12, V2, A23, V3, A34, V4, A45, V5, A56, V6, A67, V7, A71], S).

A12 = 6,
A23 = 2,

A34 = 12,
A45 = 9,
A56 = 10,
A67 = 11,
A71 = 13,
S = 21,
V1 = 1,
V2 = 14,
V3 = 5,
V4 = 4,
V5 = 8,
V6 = 3,
V7 = 7 ?

SOLUCIÓN:

El siguiente código Ciao Prolog basado en restricciones en dominios finitos resuelve el problema planteado:

```
:- use_package(fd).

resolver_heptagono([V1, A12, V2, A23, V3, A34, V4, A45, V5, A56, V6, A67, V7, A71], S) :-
        [V1, A12, V2, A23, V3, A34, V4, A45, V5, A56, V6, A67, V7, A71] in 1..14,
        all_different([V1, A12, V2, A23, V3, A34, V4, A45, V5, A56, V6, A67, V7, A71]),
        V1 + A12 + V2 .=. S,
        V2 + A23 + V3 .=. S,
        V3 + A34 + V4 .=. S,
        V4 + A45 + V5 .=. S,
        V5 + A56 + V6 .=. S,
        V6 + A67 + V7 .=. S,
        V7 + A71 + V1 .=. S,
        labeling([V1, A12, V2, A23, V3, A34, V4, A45, V5, A56, V6, A67, V7, A71, S]).
```

Solución alternativa, utilizando la sintaxis de SWI Prolog para dominios finitos:

```
:- use_module(library(clpfd)).
:- set_prolog_flag(toplevel_print_options, [quoted(true), portray(true)]). % Forzar que SWI prolog no "resuma" la solución

resolver_heptagono([V1, A12, V2, A23, V3, A34, V4, A45, V5, A56, V6, A67, V7, A71], S) :-
        [V1, A12, V2, A23, V3, A34, V4, A45, V5, A56, V6, A67, V7, A71] ins 1..14,
        all_different([V1, A12, V2, A23, V3, A34, V4, A45, V5, A56, V6, A67, V7, A71]),
        V1 + A12 + V2 #= S,
        V2 + A23 + V3 #= S,
        V3 + A34 + V4 #= S,
        V4 + A45 + V5 #= S,
        V5 + A56 + V6 #= S,
        V6 + A67 + V7 #= S,
        V7 + A71 + V1 #= S,
        label([V1, A12, V2, A23, V3, A34, V4, A45, V5, A56, V6, A67, V7, A71, S]).
```