PROGRAMACIÓN DECLARATIVA: LÓGICA Y RESTRICCIONES

Grado en Ingeniería Informática / Grado en Matemáticas e Informática

9 de Julio de 2015 **Examen Extraordinario**

Nombre:

Matrícula:

INSTRUCCIONES: El examen consta de 3 ejercicios. Todos los ejercicios deben comenzar a contestarse en su hoja correspondiente. Pueden añadirse a los ejercicios 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: Programación Lógica Pura (3,5 puntos)

Se pide programar **dos predicados lógicos puros** para las operaciones de inserción y eliminación en árboles binarios, operaciones que son una inversa de la otra. Los predicados se especifican como sigue. Los árboles se representan de la forma habitual. El camino en un árbol hasta un nodo n (no necesariamente una hoja) como una lista con los nodos que recorre, en el orden en que se recorren, empezando por el nodo raíz del árbol y finalizando en n. El argumento C más abajo es un camino que finaliza con el nodo raíz del subárbol en el que insertar/eliminar el elemento; a este subárbol se le denomina pivote.

El predicado **insertar**(A0,E,C,A1) es cierto si y solo si A1 es el árbol resultante de insertar el elemento E en el árbol binario (no vacío) A0 en la posición indicada por el camino C. El nuevo elemento se incluye como hijo izquierdo del subárbol pivote. Si el pivote ya tiene hijo izquierdo, éste (el mencionado hijo izquierdo) se pone como hijo derecho. Si el pivote tiene hijo derecho, éste (el mencionado hijo derecho) se pone como hijo derecho del subárbol del nuevo nodo insertado. En ningún caso el subárbol insertado tiene hijo izquierdo.

El predicado **eliminar**(A0,C,E,A1) es cierto si y solo si A1 es el árbol (no vacío) resultante de eliminar el elemento E del árbol binario A0 en la posición indicada por el camino C. El elemento eliminado es nodo raíz del hijo izquierdo del subárbol pivote. Si el pivote tiene hijo derecho, éste (el mencionado hijo derecho) se pone como hijo izquierdo. Si el subárbol eliminado tiene hijo derecho, éste (el mencionado hijo derecho) se pone como hijo derecho del subárbol pivote. Se supone que el subárbol eliminado no tiene hijo izquierdo.

```
insertar(tree(N,I,D), E, [N|Ns], tree(N,NewI,NewD)):-
        New = tree(E,void,X),
        llegar_al_pivote(Ns, I, D, NewI, NewD, New, X).

llegar_al_pivote([], I, D, New, I, New, D).
llegar_al_pivote([N|Ns], tree(N,I,D), DD, tree(N,NI,ND), DD, New, X):-
        llegar_al_pivote(Ns, I, D, NI, ND, New, X).
llegar_al_pivote([N|Ns], II, tree(N,I,D), II, tree(N,NI,ND), New, X):-
        llegar_al_pivote(Ns, I ,D, NI, ND, New, X).

eliminar(A0,C,E,A1):- insertar(A1,E,C,A0).
```

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

9 de Julio de 2015 **Examen Extraordinario**

Nombre:

Matrícula:

INSTRUCCIONES: El examen consta de 3 ejercicios. Todos los ejercicios deben comenzar a contestarse en su hoja correspondiente. Pueden añadirse a los ejercicios 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 2: ISO-Prolog (3,5 puntos)

Definir el predicado **residuo/3** (residuo(+L1,+L2,-R)) que se verifica si R es la lista de los elementos de L1 que no están en L2. Debe generar solamente una solución y **no** se puede usar negación. Además, codificar una llamada a dicho predicado que obtenga una lista como la mencionada R pero sin redundancias.

Definir el predicado **operacionAritmetica/4** (operacionAritmetica(+Op,+L1,+L2,-L3)) que se verifica si L3 es la lista que se obtiene al aplicar la operación aritmética (binaria) Op a los elementos de L1 y L2 que ocupan la misma posición. Se supone que (a) L1 y L2 son listas de números de la misma longitud y (b) Op es una operación binaria predefinida del lenguaje Prolog.

Definir el predicado **soloPares/2** (soloPares(+L,-LPares)) que se verifica si LPares es la lista de los elementos pares de L (en el mismo orden en el que aparecen en L). El código **no** debe realizar recorridos de listas.

```
residuo([],_,[]) :- !.
residuo([X|Xs],L2,R) :-
        member(X,L2),!,
        residuo(Xs,L2,R).
residuo([X|Xs],L2,[X|R]) :-
        residuo(Xs,L2,R).

% Llamada sin redundancias:  setof(E,(residuo(L1,L2,Rr),member(E,Rr)),R).

operacionAritmetica(_,[],[],[]).
operacionAritmetica(Op,[X|Xs],[Y|Ys],[R|Rs]):-
        Exp =.. [Op,X,Y],
        R is Exp,
        operacionAritmetica(Op,Xs,Ys,Rs).

soloPares(L,LPares):-
        findall(E,(member(E,L),par(E)),LPares).
par(N):-
        number(N),
        N mod 2 =:= 0.
```

PROGRAMACIÓN DECLARATIVA: LÓGICA Y RESTRICCIONES

Grado en Ingeniería Informática / Grado en Matemáticas e Informática

9 de Julio de 2015 **Examen Extraordinario**

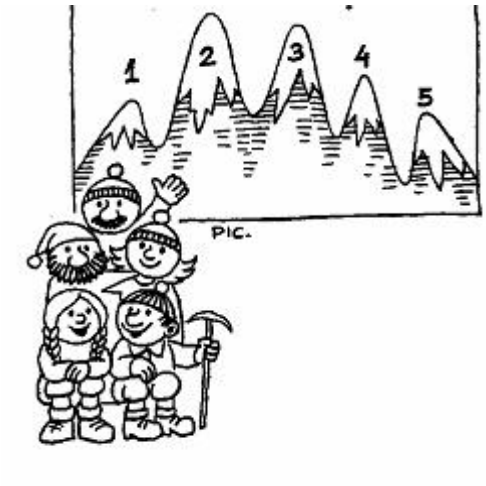
Nombre:

Matrícula:

INSTRUCCIONES: El examen consta de 3 ejercicios. Todos los ejercicios deben comenzar a contestarse en su hoja correspondiente. Pueden añadirse a los ejercicios 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: Programación Lógica con Restricciones (3 puntos)



LOS PICOS DE ESTA CORDILLERA SE LLAMAN COMO LOS MONTAÑEROS QUE LOS HAN CONQUISTADO.
JOSÉ ESTÁ ENTRE CARLOTA Y MARÍA.
ANTONIO ESTÁ ENTRE CARLOTA Y RAMÓN.
LOS MÁS ALTOS SON JOSÉ Y CARLOTA.
¿CÓMO SE LLAMA CADA PICO?

Se pide al alumno que escriba un **programa lógico con restricciones en dominios finitos** con la sintaxis de **Ciao Prolog** que calcule todas las soluciones existentes para este problema.

Un posible enfoque para resolver el problema es representar cada pico mediante una variable cuyo nombre coincide con el nombre de dicho pico, es decir: José, Carlota, María, Antonio y Ramón. El dominio asociado a dichas variables será discreto y finito: {1, 2, 3, 4, 5}. Siguiendo este enfoque, el programa a desarrollar asignará a cada variable un valor diferente de dicho dominio respetando las restricciones del enunciado:

```
:- module(_,_,[fd]).
picos([Jose,Carlota,Maria,Antonio,Ramon]) :-
        [Jose,Carlota,Maria,Antonio,Ramon] in 1..5,
        all_different([Jose,Carlota,Maria,Antonio,Ramon]),
        (
                Jose - 1 .=. Carlota, Jose + 1 .=. Maria ;
                Jose - 1 .=. Maria, Jose + 1 .=. Carlota
        ),
        (
                Antonio - 1 .=. Carlota, Antonio + 1 .=. Ramon ;
                Antonio - 1 .=. Ramon, Antonio + 1 .=. Carlota
        ),
        (
                Jose .=. 2, Carlota .=. 3 ;
                Carlota .=. 2, Jose .=. 3
        ),
        labeling([Jose,Carlota,Maria,Antonio,Ramon]).
```

Para resolver el problema, se hace la siguiente pregunta al interprete de Prolog:

```
?- picos([Jose,Carlota,Maria,Antonio,Ramon]).
Antonio = 4,
Carlota = 3,
Jose = 2,
Maria = 1,
Ramon = 5 ? ;
no
?-
```

Que como puede verse, arroja una solución única al problema: los picos son, de izquierda a derecha según se muestra en la figura: María, José, Carlota, Antonio y Ramón.