Programación Declarativa. Grado Matemáticas. UCM.
HOJA 3 DE EJERCICIOS

1. Suponiendo que `progenitor(X,Y)` se cumple si `X` es progenitor de `Y`, mientras que `hombre(X)` y `mujer(X)` se cumplen cuando `X` es respectivamente hombre o mujer, define predicados Prolog adecuados para las siguientes relaciones familiares binarias:

   a) Ser padre de.

   b) Ser madre de.

   c) Ser hermanos.

   d) Ser tío/a de.

   e) Ser primo/as.

   f) Ser antepasado de.

   g) Ser descendiente de.

   h) Ser parientes.

2. Suponiendo que `arista(X,Y)` se cumple si hay una arista que va del nodo `X` al `Y`, y suponiendo que el grafo dado sea acíclico, define un predicado binario en Prolog que se cumpla sii existe un camino entre los dos nodos de entrada que se le pasen. ¿Funcionaría el predicado definido si el grafo contiene ciclos?

3. Repite el ejercicio anterior pero definiendo un predicado ternario donde la tercera componente sea la lista de nodos intermedios del camino que une a los nodos `X` e `Y`.

4. Calcula el unificador más general para cada una de las siguientes parejas de términos:

   a) p(X,f(Y)) y p(Z,X)

   b) f(X,g(X)) y f(g(Z),Y)

   c) f(X,g(X),g(X,Y)) y f(g(Z),g(W),g(W,W))

   d) f(g(a),X,g(X)) y f(Y,g(h),Y)

5. Define reglas Prolog adecuadas para las siguientes relaciones binarias entre listas:

   a) Ser lista inversa.

   b) Ser prefijo.

   c) Ser sufijo.

6. Define reglas Prolog adecuadas para las siguientes relaciones entre elementos y listas:

   a) Pertenecer a una lista.

   b) La segunda lista es equivalente a eliminar de la primera lista todas las apariciones del elemento dado.

   c) La segunda lista es equivalente a eliminar de la primera lista la primera aparición del elemento dado.

7. Define un predicado binario Prolog `sinDuplicados` donde la segunda lista del predicado sea igual a la primera pero eliminando de ella los elementos repetidos adyacentes. Por ejemplo, se cumpliría el predicado `sinDuplicados([a,b,b,a,c,c],[a,b,a,c])`.

8. Define predicados Prolog que se correspondan con el operador `++` y la función `concat` de Haskell.

9. Suponiendo que usamos árboles binarios donde el árbol vacío se representa como `aVacio` y los árboles no vacíos se representan como `nodo(Elemento,Iz,Dr)`, define predicados para los recorridos en preorden, inorden y postorden de un árbol binario.

10. Suponiendo que usamos números naturales donde el 0 se representa como `cero` y el sucesor de un número `n` se representa mediante `s(n)`, define predicados Prolog que se correspondan con las funciones Haskell take, drop y splitAt.

11. Repite el ejercicio anterior usando la aritmética predefinida de Prolog en lugar de usar `cero` y `s(n)`.

12. Escribe un predicado Prolog que calcule el máximo común divisor de dos naturales.

13. Repite el ejercicio 2 pero utilizando aristas con costes asociados. El predicado que determina si existe un camino entre dos nodos deberá tener una última componente en la que se incluya el coste del camino.

14. Escribe predicados Prolog para calcular el factorial de un numero, para calcular el sumatorio de los elementos de una lista, para calcular el máximo elemento de una lista, para calcular el producto escalar de los elementos de dos listas, y para calcular la suma de dos matrices representadas como listas de listas.

15. Escribe un predicado Prolog para ordenar una lista utilizando el algoritmo quicksort.

16. Escribe predicados Prolog para buscar, para añadir y para eliminar un elemento de un arbol binario de búsqueda.