Guía de Trabajos Prácticos X – Programación Lógica II

1) Cree un programa en Prolog que permita calcular el factorial de un número con el predicado "factorial/2" en donde el primer argumento unifique con el número al que se le calculará el factorial, y el segundo con el factorial del mismo.

Se debe verificar que el número pasado como primer parámetro sea mayor a cero, en caso contrario, el predicado debe fallar.

Ej.: factorial(5, X). => X = 120.

2) Escribir un programa en Prolog "contar/3" que reciba como primer parámetro una lista de números, y unifique el segundo argumento con la cantidad de elementos pares que encuentre (tomando el 0 como par) y el tercer argumento con la cantidad de elementos impares de la misma.

Si algún elemento de al lista no fuera un número entero el predicado debería fallar, para esto se puede utilizar el predicado predefinido "integer/1" que unifica solo si su argumento es un número entero.

Ejemplos:

contar([1, 2, 3], P, I).
P = 1
I = 2

contar([1, 2, a], P, I).
false

3) Escribir un predicado en Prolog que sume los elementos de una lista de números enteros recibidos como primer argumento y unifique el resultado de dicha suma con el segundo argumento.

Tener en cuenta

Ejemplos:

sumar([1, 2, 3], X).
X = 6

sumar([1, -2, 3], X).
X = 2

4) Escribir un programa en Prolog que aplane una lista. El predicado aplanar/2 recibe una lista cuyos elementos pueden ser otras listas.

No se deben utilizar predicados predefinidos si los hubiere.

Ejemplos:

aplanar([1, 3, 2], L).
L = [1, 2, 3].

aplanar([1, [3, 2]], L).
L = [1, 3, 2].

aplanar([1, [3, 2], [1, 5, 4], 3, [5, 1]], L).
L = [1, 3, 2, 1, 5, 4, 3, 5, 1].

5) Escribir un programa en Prolog que reciba dos listas de números, verifiquen que sean de la misma longitud, y luego retorne una lista con la suma elemento a elemento de ambas listas.

Ejemplos:

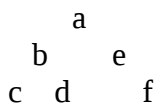
sumar([1, 2], [1], L). %falla porque son de distinto tamaño
false

sumar([1, 2], [5, 3], L).
L = [6, 5]

6) Escribir un programa en Prolog que recorra un árbol binario y determine la profundidad del mismo.

La representación del árbol será una lista con el siguiente formato: [I, N, D] en donde:
I es una lista que representa el subárbol de la rama izquierda
N es el valor del nodo raíz
D es una lista que representa el subárbol de la rama derecha

así el árbol:



estaría representado por [[[c], b, [d]], a, [[], e, [f]]]

prestar atención a que el las ramas vacías se representan con una lista vacía, y las hojas como un alista de un solo elemento.

7) Tomando en cuenta las consideraciones del ejercicio anterior, realizar un predicado que cuente por nivel del árbol la amplitud, y retorne la mayor amplitud encontrada en el mismo.

8) Desarrollar un programa en Prolog que permita calcular las permutaciones de una lista.

Ej.: permutaciones([a, b, c], P).
P=[a,b,c];
P=[a,c,b];
P=[b,a,c];…

9) Crear un programa en Prolog que dado un tablero de ajedrez, permita colocar tantas reinas como se quiera en el mismo, para que luego dada una posición nos informe que reinas la acechan.

El programa debe prever que la posición a analizar no debe estar ocupada por una reina, de ser así el programa debe fallar.

Otro punto a tener en cuenta es que las posiciones de la reina y de la posición a verificar deben ser validadas para no permitir que se salgan del tablero (8x8)

Las reinas que participarán deberán incorporarse como una lista, donde a su vez, cada elemento representando una reina, será una lista de dos elementos (X, Y).

La declaración de las reinas se da mediante el predicado “reinas/1”, por ejemplo:
reinas([[3, 4], [5, 4], [5, 3]]).

La posición a validar con el predicado “validar/3” por ejemplo:
validar(6, 7, X).

El tercer parámetro deberá unificar con una lista que contendrá las posiciones de las reinas que acechen la casilla en cuestión, en caso de no haber reinas que la acechen se deberá unificar con una lista vacía.