

Guía de Trabajos Prácticos IX – Programación Lógica II

- 1) Cree un programa en Prolog que permita calcular el factorial de un número con el predicado “factorial/2”, validando que dicho número sea mayor o igual a cero.

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

- 2) Cree un programa en Prolog que cuente la cantidad de veces que aparece un elemento en una lista.

Ej.: contar(y, [a, b, c, a, d, e, a, f, a], Cantidad). => Cantidad = 0.

Ej.: contar(a, [a, b, c, a, d, e, a, f, a], Cantidad). => Cantidad = 4.

- 3) Escribir un programa en Prolog “contar/3” que reciba como primer parámetro una lista de números y unifique el segundo con la cantidad de elementos de dicha lista.

Ej.: cantidad([a, b, c], Elementos). => Elementos = 3.

- 4) Escriba un programa en Prolog que dada una lista de números enteros, calcule el resultado de sumar dichos números.

Ej.: suma([1, 2, 3], X). => X = 6.

- 5) Escriba un programa en Prolog que dada una lista de números enteros, retorne otra lista solo con los números positivos de la misma.

Ej.: positivos([1, -2, 3, -4], ListaPositivos). => ListaPositivos = [1, 3].

- 6) 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.

Ej.: suma_lista([1, -2, 3, -4], [2, 3, 1, 4], ListaSuma). => ListaSuma = [3, 1, 4, 0].

- 7) Escriba un programa en Prolog que dada una lista elimine todos los elementos duplicados de la misma.

Ej.: eliminar_dup([1, 2, 3, 1, 4, 3, 5, 6], SinDup). => SinDup = [1, 2, 3, 4, 5, 6].

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

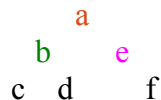
La representación del árbol será una lista con el siguiente formato: [I, N, D] en donde:

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]]]`

Las ramas vacías se representan con una lista vacía, y las hojas como una lista de un solo elemento.

Ej.: `profundidad([[[c], b, [d]], a, [[], e, [f]]], Profundidad).` => `Profundidad = 3.`

- 9) Escriba un programa en Prolog que dada una lista numérica ordenada, inserte un elemento en el lugar correspondiente según el orden.

Ej.: `insertar(3, [1, 2, 4, 5], Resultado).` => `Resultado = [1, 2, 3, 4, 5].`

- 10) Escriba un programa en Prolog que recursivamente ordene una lista de números enteros.

Ej.: `ordenar([2, 4, 3, 1], ListaOrdenada).` => `ListaOrdenada = [1, 2, 3, 4].`

- 11) Escribir un programa en Prolog que aplane una lista. El predicado `aplanar/2` recibe una lista cuyos elementos pueden ser otras listas y debe retornar una lista con todos los elementos atómicos presentes.

Ej.: `aplanar([1, 2, 3], ListaPlana).` => `ListaPlana = [1, 2, 3]`

Ej.: `aplanar([1, 2, [3]], ListaPlana).` => `ListaPlana = [1, 2, 3]`

Ej.: `aplanar([1, [2, [3]]], ListaPlana).` => `ListaPlana = [1, 2, 3]`

- 12) El siguiente programa en Prolog calcula las permutaciones de los elementos de una lista.

a) Ejecute el mismo y escriba el resultado obtenido para `per([1, 2, 3], L).`

b) Explique en sus propios términos cual es la lógica que utiliza el programa para obtener las permutaciones.

```
ins(X, L, [ X | L ]).
```

```
ins(X, [ Y | L1 ], [ Y | L2 ]) :- ins(X, L1, L2).
```

```
per([], []).
```

```
per([ X | L ], Lp) :- per(L, L1), ins(X, L1, Lp).
```