1)IMPLEMENTAR: Recursivo que calcula la cantidad de veces que un elemento está en una lista

ejemplo 1: ocurrencias(2, [3,5,8,6,2,3,2,3,2], N) el resultado es N = 3

ejemplo 2: ocurrencias(s, [d,a,d,g,g,s,d,f], N) el resultado es N = 1

Las reglas que debe usar son estas:

ocurrencias(_,[],0).

ocurrencias(Cabeza,[Cabeza|Cola],N) :-

ocurrencias(Elemento,[Cabeza|Cola],N) :-

2)

2.1) IMPLEMENTAR: Recursivo que cuenta la cantidad de ítems que contiene una lista

ejemplo 1: cantidad([3,5,8,6,2,3,2,3,2], N) el resultado es N = 9

ejemplo 2: cantidad([d,a,d,g,g,s,d,f], N) el resultado es N = 8

Las reglas que debe usar son estas:

cantidad([],0).

cantidad([_|Cola],Total) :-

2.2) IMPLEMENTAR: Recursivo que quita todas las ocurrencias de un elemento en una lista

ejemplo 1: quitar(2, [3,5,8,6,2,3,2,3,2], N) el resultado es N = [3,5,8,6,3,3]

ejemplo 2: quitar(g, [d,a,d,g,g,s,d,f], N) el resultado es N = [d,a,d,s,d,f]

Las reglas a usar son las siguientes:

quitar(_,[],[]).

quitar(Cabeza, [Cabeza|Cola], Res) :-

quitar(Elemento, [Cabeza|Cola], Res) :-

3) IMPLEMENTAR: El podado(no, el que piensas no es)

Es una técnica para limpiar las componentes de una red neuronal,

una técnica de podado consiste en quitar según algún índice o métrica unos elementos

de un vector o matriz (Lista).

En un sentido simple, implemente una función para Quitar todos los elementos

cuyo índice de ocurrencia o métrica de incidencia

(= Ocurrencia del Elemento en la Lista / Total Inicial de Elementos Lista)

es mayor que 0.25

ejemplo: podar([ok,tv, radio, tv, play, tv, tv, radio],8,Resp) el resultado es Resp = [ok, play]

ejemplo: podar([5,5,7,4,1,2,4,7,8,5,1,2,1,1,2,1,2],17,Resp) el resultado es

Resp = [5, 5, 7, 4, 2, 4, 7, 8, 5, 2, 2, 2]

Las reglas a usar son las siguientes:

```
podar([],_,[]):-!.

podar(Lista, Ncant, LRes):-

    /*06. TU CODIGO VA ACÁ, cuando es < 0.25*/

podar(Lista, Ncant, LRes):-

    /*07. TU CODIGO VA ACÁ, cuando es >= 0.25*/).
```

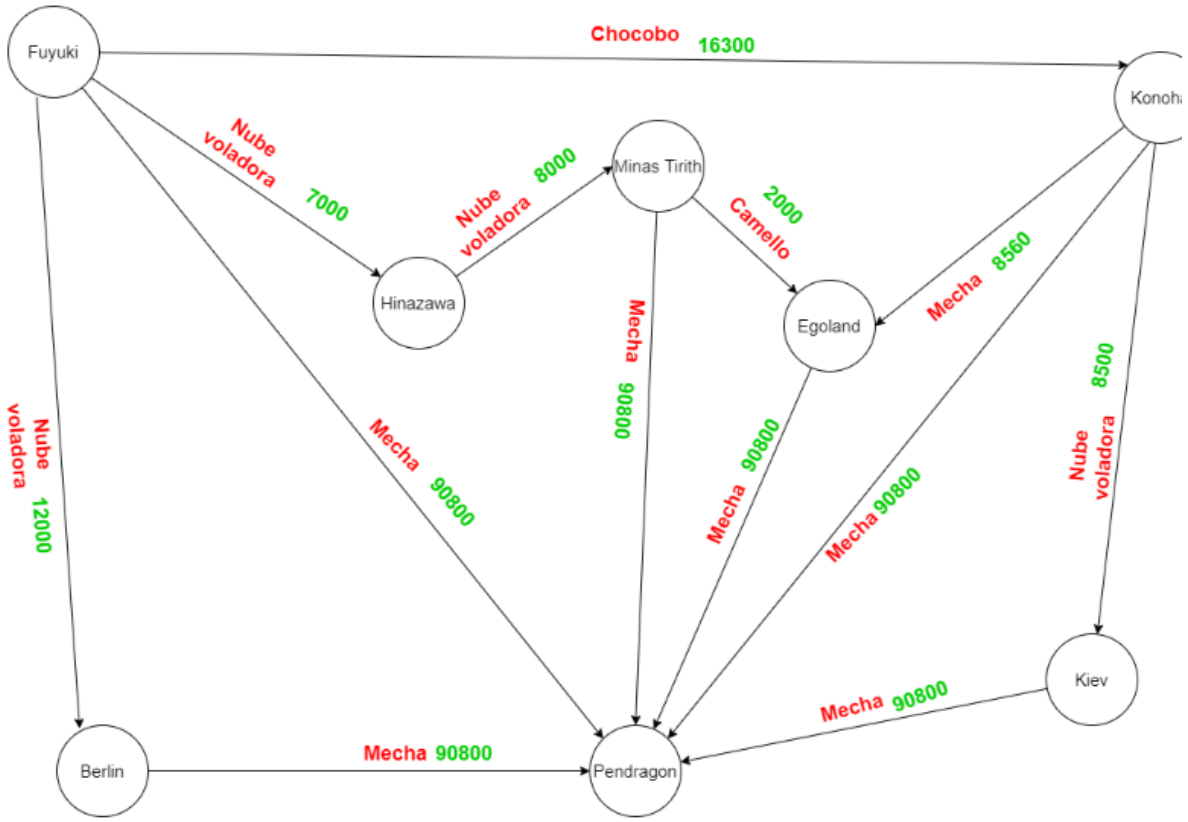
4) Tenemos un programa llamado viajeros, donde tenemos los siguientes hechos:

```
link(fuyuki, konoha, chocobo, 16300).
link(fuyuki,hinazawa,nube_voladora,7000).
link(fuyuki,berlin,nube_voladora,12000).
link(fuyuki,pendragon,mecha,90800).
link(konoha,egoland,mecha,8560).
link(konoha,kiev,nube_voladora,8500).
link(konoha,pendragon,mecha,90800).
link(hinazawa,minas_tirith,nube_voladora,8000).
link(minas_tirith,egoland,camello,2000).
link(minas_tirith,pendragon,mecha,90800).
link(egoland,pendragon,mecha,90800).
```

link(kiev,pendragon,mecha,90800).
link(berlin,pendragon,mecha,90800).

Como podemos observar tenemos ciudad origen, ciudad destino, transporte y precio del trayecto.

GRAFO QUE DESCRIBE LOS HECHOS:



Ustedes tienen que completar los hechos y las reglas del programa viajeros, de tal forma que se generen las listas con la información del trayecto como aparece en el archivo adjunto. De esta forma, debe aparecer Ciudades (Lista de ciudades en el recorrido), Transportes (Lista de medios de transporte correspondientes), Precios (Lista con el costo de viaje en cada trayecto). También, debe dar el Valor Total del Viaje a la ciudad de Destino.

enrutar(Ciudad_Ini, Ciudad_Ter, M, Pp , P, T) :- ...

Por ejemplo, uno de los trayectos posibles, entre fuyuki y egoland, puede ser:

Ciudad_Ini = fuyuki,

Ciudad_Ter = egoland,

M = [fuyuki, konoha, egoland],

P = 24860,

Pp = [16300, 8560],

T = [chocobo, mecha]

NOTA: Las variables M (ciudades del trayecto), P (Precio total del trayecto), Pp (Lista de precios entre ciudades) , T (lista de transportes), SON LAS VARIABLES que usted debe RETORNAR la lista con sus posibles valores.

Ejercicios propuestos por el monitor:

5) En este ejercicio se le entrega la siguiente lista:

Se le pide implementar otra regla para contar los elementos de una lista(Incluida las sub listas de sublistas y asi sucesivamente), pero para una lista 3x3x3(regla size)

size([],R,R).
size([_|T],Ac,R) :-

También se le pide sumar los elementos de una lista 3x3x3
(Incluida las sub listas de sublistas y asi sucesivamente)(regla iterate)

iterate([],R,R).
iterate([H|T],Ac,R):-

Para al final utilizar las funciones creadas en el vpl para calcular el promedio de los elementos de la lista (como el ejercicio en clase,Incluida las sub listas de sublistas y asi susecivamente)3x3x3:

promedio(X,R):-

6) Dada la siguiente fracción continua, debe realizar una regla utilizando tail recursión para encontrar la n-esima fracción continua dada.

$$1-\cfrac{x^1}{3-\cfrac{x^2}{5-\cfrac{x^3}{7-\cfrac{x^4}{9-\cfrac{x^5}{\ddots\ (2*n-1)-\cfrac{x^n}{2*n+1}}}}}}$$

NOTA: frac1(A,B,C) en C se manda la respuesta, tanto A como B la suministra el sistema.

A es n y B es x de la fracción continua

7) Resolver mediante verificación de restricciones en PROLOG el Sudoku

4*4 que aparece a continuación. Dar al menos dos soluciones

		1	
	1		2
			3
		4	

% HECHOS

num(1). num(2). num(3). num(4).

% REGLAS

unicos(P,Q,R,S) :- num(P), num(Q), num(R), num(S),

\+ P=Q, \+ P=R, \+ P=S, \+ Q=R, \+ Q=S, \+ R=S.

sudoku(R11,R12,R13,R14,

R21,R22,R23,R24,

R31,R32,R33,R34,

R41,R42,R43,R44) :-

% Completar aquí …

Restricciones Iniciales:

1. Los números en cada fila son únicos (1,2,3,4).

2. Los números en cada columna son únicos

(1,2,3,4).

3. Los cuadrantes 2X2 (NO,NE,SO,SE) son únicos