

%

% CODIGO PARA OPERACIONES CON LISTAS

**% GENERALIDADES**

% Determinar si lo que se recibe es una lista

% Ejemplo: lista(1). No  
% lista([1,2]). Yes

lista([]):-!.  
lista([\_X|Y]):-lista(Y).

% Devuelve la longitud de la lista que se le pasa por argumento.

% Ejemplo: long([1,3,7,4],X). X=4

long([],0):-!.  
long([\_X|Y],S):-long(Y,T), S is T+1.

% Maximo elemento de una lista

% Ejemplo: maximo([1,5,3,-2],X). X=5

maximo([X],X).  
maximo([X|Xs],X):-maximo(Xs,Y), X >= Y.  
maximo([X|Xs],N):-maximo(Xs,N), N > X.

% Minimo elemento de una lista

% Ejemplo: minimo([1,5,3,-2],X). X=-2

minimo([X],X).  
minimo([X|Xs],X):-minimo(Xs,Y), X =< Y.  
minimo([X|Xs],N):-minimo(Xs,N), N < X.

% Determina si una lista es estrictamente creciente

% Ejemplo: crece([1,2,3]). Yes  
% crece([2,2,3]). No

crece([\_X]).  
crece([X,Y|Z]):-X<Y, crece([Y|Z]).

% Determina si una lista es estrictamente decreciente

% Ejemplo: decrece([3,2,1]). Yes  
% decrece([3,2,2]). No

decrece([\_X]).  
decrece([X,Y|Z]):-X>Y, decrece([Y|Z]).

% Determina si dos listas son iguales en todos los niveles

% Ejemplo: listaIgual([1,2,[5,2],3],[1,2,[5,2],3],9). No  
% listaIgual([1,2,[5,2],3],[1,2,[5,2],3]). Yes

listaIgual([],[]):-!.  
listaIgual([X|M],[X|R]):-listaIgual(M,R).  
listaIgual([X|M],[T|R]):-lista(X), lista(T), listaIgual(X,T), listaIgual(M,R).

% Determina si una lista es capicua

% Ejemplo: capicua([n,e,u,q,u,e,n]). Yes  
% capicua([n,e,u,q,[u,e],n]). No  
% capicua([n,e,[u,q,u],e,n]). Yes

capicua([]):-!.  
capicua(L):-invertir(L,R), listaIgual(L,R).

% Determina si la primer lista es prefijo de la segunda

% Ejemplo: prefijo([1,2,3],[1,2,3,4,5]). Yes  
% prefijo([1,5,3],[1,2,3,4,5]). No

prefijo([],\_M):-!.  
prefijo([\_X],[\_X|\_M]):-!.  
prefijo([\_X|L],[\_X|M]):-prefijo(L,M).

```

% Determina si la primer lista es posfijo de la segunda.
% Ejemplo:      posfijo([3,5,4],[1,2,6,5,7,8,3,5,4]).      Yes
%              posfijo([3,5,4],[1,2,6,5,7,8,3,6,4]).      No

```

```
posfijo(L,L1):-invertir(L,X),invertir(L1,Y),prefijo(X,Y).
```

```

% Determina si la primer lista es sublista de la segunda
% Ejemplo:      subLista([3,2],[1,6,2,3]).                  No
%              subLista([3,2],[1,6,3,2]).                  Yes

```

```

subLista([],_L):-!.
subLista(L,[X|M]):-prefijo(L,[X|M]).
subLista(L,[X|_M]):-lista(X),subLista(L,X).
subLista(L,[_X|M]):-subLista(L,M).

```

```

% Determina si dos elementos son consecutivos
% Ejemplo:      consecutivo([1,3,4,6,5,7,1,2],6,5).        Yes
%              consecutivo([1,3,4,6,5,7,1,2],5,6).        No

```

```

consecutivo([_X,_Y|_Xs],_X,_Y).
consecutivo([_X|Xs],N,Z):-consecutivo(Xs,N,Z).

```

### % BUSQUEDA DE ELEMENTOS

```

% Obtener el primer elemento de la lista
% Ejemplo:      primerElem([3,2,4],X).                      X=3

```

```
primerElem([_X|_Y],_X).
```

```

% Obtener el ultimo elemento de la lista
% Ejemplo:      ultimoElem([3,2,4],X).                      X=4

```

```
ultimoElem(L,S):-invertir(L,T),primerElem(T,S).
```

```

% Devuelve el enesimo elemento
% Ejemplo:      enesimoElem([1,4,3,2,5],2,X).               X=4

```

```

enesimoElem([],_N,[]):-!.
enesimoElem([_X|_Y],1,_X):-!.
enesimoElem([_X|Y],N,E):-N1 is N - 1,enesimoElem(Y,N1,E).

```

```
% Devuelve una lista con el elemento que se encuentra en la enesima posicion
```

```

% Ejemplo:      nPosicion([1,4,2,3],2,X).                   X=[4]
%              nPosicion([1,4,2,3],8,X).                     X=[]

```

```

nPosicion([],_N,[]):-!.
nPosicion([_X|_N],1,[X]):-!.
nPosicion([_X|R],N,S):-M is N-1,nPosicion(R,M,S).

```

```
% Devuelve el numero de posicion de la primera ocurrencia de X
```

```

% Ejemplo:      xPosicion(4,[7,2,1,8,3,6],X).              X=0
%              xPosicion(3,[7,2,1,3,4,3],X).               X=4

```

```

xPosicion(X,L,S):-pertenece(X,L),!,xBusca(X,L,S).
xPosicion(X,L,S):-not(pertenece(X,L)),S is 0.

```

```

xBusca(_X,[],0):-!.
xBusca(_X,[_X|_M],1):-!.
xBusca(X,[_Y|M],S):-xBusca(X,M,T),S is T +1.

```

```
% Determina si un elemento X pertenece a la lista L (1 nivel)
```

```

% Ejemplo:      pertenece(1,[3,4,6,1,3]).                   Yes           pertenece(1,[3,[4,6,1],3]).      No

```

```

pertenece(_X,[_X|_Y]).
pertenece(X,[_C|Y]):-pertenece(X,Y).

```

```
% Determina si un elemento X pertenece a la lista L (multinivel)
```

```

% Ejemplo:      perteneceM(1,[3,4,6,1,3]).                  Yes
%              perteneceM(1,[3,[4,6,1],3]).                 Yes

```

```
perteneceM(N,L):-listaAtomos(L,La),pertenece(N,La).
```

## % ELIMINACION O REEMPLAZOS DE ELEMENTOS EN UNA LISTA

% Elimina los N primeros elementos de una lista y devuelve el resto  
% Ejemplo: sacaNpri([1,3,5,1,2],2,L). L=[5,1,2]

```
sacaNpri([],_N,[]):-!.  
sacaNpri([_X|_M],1,_M):-!.  
sacaNpri([_X|M],N,S):-N1 is N - 1,sacaNpri(M,N1,S).
```

% Elimina los N ultimos elementos de una lista y devuelve el resto.  
% Ejemplo: sacaNult([1,2,3,4,5],2,L). L=[1,2,3]

```
sacaNult(L,N,R):-invertir(L,L1),sacaNpri(L1,N,R1),invertir(R1,R).
```

% Elimina el elemento X de la lista en el 1º nivel  
% Ejemplo: eliminaX([1,2,3,4,5],3,L). L=[1,2,4,5]  
% eliminaX([1,2,[3,4],5,3],3,L). L=[1,2,[3,4],5]

```
eliminaX([],_X,[]):-!.  
eliminaX([X|M],X,Z):-eliminaX(M,X,Z),!.  
eliminaX([R|M],X,[R|Z]):-eliminaX(M,X,Z),!.
```

% Elimina el elemento X de la lista en todos los niveles  
% Ejemplo: eliminaMx([1,2,[3,4],5,3],3,L). L=[1,2,[4],5]  
% eliminaMx([2,[3,4],5,3],4,L). L=[2,[3],5,3]

```
eliminaMx([],_X,[]):-!.  
eliminaMx([X],X,[]):-!.  
eliminaMx([X|M],X,S):-eliminaMx(M,X,S),!.  
eliminaMx([R|M],X,S):-lista(R),eliminaMx(R,X,T),eliminaMx(M,X,P),concatenar([T],P,S),!.  
eliminaMx([R|M],X,S):-eliminaMx(M,X,T),concatenar([R],T,S).
```

% Elimina todos los elementos de la lista 2 que estan en la 1  
% Ejemplo: elim12([1,4,3,2,7,9],[2,1],L). L=[]  
% elim12([2,1],[1,4,3,2,7,9],L). L=[4,3,7,9]  
% elim12([1,4],[1,2,[4,6],5,3],L). L=[2,[6],5,3]

```
elim12([],L,L):-!.  
elim12([X|M],L,S):-eliminaMx(L,X,T),elim12(M,T,S).
```

% Elimina los elementos repetidos que estan en una lista, dejando el que primero ocurre  
% Ejemplo: eliminaR([2,[3,4],5,3],L). L=[2,[3,4],5]  
% eliminaR([2,[3,4],1,5,1,3],L). L=[2,[3,4],1,5]

```
eliminaR([],[]):-!.  
eliminaR([X|M],S):-not(lista(X)),eliminaX(M,X,T),eliminaR(T,Y),concatenar([X],Y,S).  
eliminaR([X|M],S):-lista(X),elim12(X,M,T),eliminaR(X,Y),  
eliminaR(T,J),concatenar([Y],J,S).
```

% Elimina el primer elemento X que aparece en la lista.  
% Ejemplo: eliminarPri(2,[4,2,1,2,1],X). X=[4,1,2,1]  
% eliminarPri(3,[4,2,1,3,1],X). X=[4,2,1,1]

```
eliminarPri(_X,[],[]):-!.  
eliminarPri(_X,[_X|_M],_M):-!.  
eliminarPri(X,[_R|M],[_R|L]):-eliminarPri(X,M,L).
```

% Elimina el elemento que se encuentra en la enesima posicion  
% Ejemplo: xElimina([1,2,3],2,L). L=[1,3]  
% xElimina([1,2,3],5,L). No

```
xElimina([_X|_Y],1,_Y):-!.  
xElimina([_X|Y],N,[_X|R]):-N1 is N - 1, xElimina(Y,N1,R).
```

% insertar X en la posicion N dentro de una Lista. N (L,N,X,R)  
% Ejemplo: insertarXenN([1,2,3,4],1,5,R). R=[5,2,3,4]

```
insertarXenN([_C|L],1,X,[X|L]):-!.  
insertarXenN([C|L],N,X,[C|R]):-N1 is N-1, insertarXenN(L,N1,X,R).
```

% Reemplaza la aparicion de un elemento X en una lista, en todos los niveles, por otro elemento Y.

% Ejemplo: xReemplazar(1,2,[1,2,1,4],X). X=[2,2,2,4]  
% xReemplazar(1,2,[1,7,[1,3,6,9],1,4],X). X=[2,7,[2,3,6,9],2,4]

```
xReemplazar(_X,_Y,[],[]):-!.  
xReemplazar(X,Y,[X|M],[Y|Z]):-xReemplazar(X,Y,M,Z),!.  
xReemplazar(X,Y,[L|M],Z):-  
xReemplazar(X,Y,L,T),xReemplazar(X,Y,M,R),!,concatenar([T],R,Z).  
xReemplazar(X,Y,[L|M],[L|Z]):-xReemplazar(X,Y,M,Z),!.
```

% Devuelve la lista original sin el ultimo elemento

% Ejemplo: qu([1,2,3,4],L). L=[1,2,3]

```
qu(L,S):-long(L,Long),Aux is Long - 1, nPrimeros(L,Aux,S).
```

% Devuelve la lista original sin el primer elemento.

% Ejemplo: qp([1,2,3],L). L=[2,3]

```
qp(L,S):-sacaNpri(L,1,S).
```

## % CREACION DE LISTAS

% concatenar dos listas

% Ejemplo: concatenar([1,2],[3,4],X). X=[1,2,3,4]

```
concatenar([],L,L):-!.  
concatenar([X|M],L,[X|Y]):-concatenar(M,L,Y).
```

% concatena un Elemento a una Lista

% Ejemplo: concatenarElem([1,3,4],2,L). L=[1,3,4,2]

```
concatenarElem([],L,[L]).  
concatenarElem([X|Xs],Y,[X|K]):-concatenarElem(Xs,Y,K).
```

% Devuelve todas las sublistas posibles en L de la lista pasada como primer argumento.

% Ejemplo: subSec([1,2,4],L). L=[1] [1,2] [1,2,4] [2] [2,4] [4]

```
subSec(M,L):-prefijo2(L,M).  
subSec([_X|M],L):-subSec(M,L).
```

```
prefijo2([_X],[_X|_L]).  
prefijo2([_X|Xs],[_X|L]):-prefijo2(Xs,L).
```

% Lista Atomos: genera una lista de atomos a partir de una lista anidada

% Ejemplo: listaAtomos([7,2,[casa,1],a],L). L=[7,2,casa,1,a]

```
atomo(X):-not(lista(X)).  
listaAtomos([],[]).  
listaAtomos([X|Y],[X|Ys]):-atomo(X),listaAtomos(Y,Ys).  
listaAtomos([X|Y],Ys):-not(atomo(X)),listaAtomos(X,AtomosX),  
concatenar(AtomosX,Y,Z),listaAtomos(Z,Ys).
```

% Da como resultado los N primeros elementos de una lista

% Ejemplo: nPrimeros([1,2,3,4,5],2,L). L=[1,2]  
% nPrimeros([1,2,[4,6],5,3],3,L). L=[1,2,[4,6]]

```
nPrimeros(_L,0,[]):-!.  
nPrimeros([],_N,[]):-!.  
nPrimeros([_X|_M],1,[_X]):-!.  
nPrimeros([X|M],N,S):-N1 is N - 1,nPrimeros(M,N1,T),concatenar([X],T,S).
```

% Devuelve todos los resultados posibles

% Ejemplo: nPrimerosT([1,2,3],L). L=[1] [1,2] [1,2,3]

```
nPrimerosT([_X|_M],[_X]).  
nPrimerosT([X|M],L):-nPrimerosT(M,T),concatenar([X],T,L).
```

```

% Da como resultado los N ultimos elementos de una lista
% Ejemplo:      nUltimos([1,2,3,4,5],2,L).      L=[4,5]
%              nUltimos([1,2,[4,6],5,3],3,L).  L=[[4,6],5,3]

nUltimos(L,N,S):-invertir(L,T),nPrimeros(T,N,R),invertir(R,S).

% Arma una lista con todos los elementos menores que X
% Ejemplo:      xMenores(3,[2,3,1,7,0],L).      L=[2,1,0]

xMenores(_X,[],):-!.
xMenores(X,[Y|W],[Y|Z]):-X>Y,xMenores(X,W,Z),!.
xMenores(X,[_Y|W],Z):-xMenores(X,W,Z),!.

% Arma una lista con todos los elementos mayores que X
% Ejemplo:      xMayores(3,[2,3,1,7,0],L).      L=[7]

xMayores(_X,[],):-!.
xMayores(X,[Y|W],[Y|Z]):-X<Y,xMayores(X,W,Z),!.
xMayores(X,[_Y|W],Z):-xMayores(X,W,Z),!.

% Devuelve una lista con los anteriores elementos a X (primera ocurrencia)
% Ejemplo:      ant(4,[3,2,1,4,1,6],L).      X=[3,2,1]
%              ant(6,[5,6,9,4,8,7],L).      X=[5]

ant(_X,[],):-!.
ant(_X,[_X|_Y],):-!.
ant(X,[C|Y],[C|L]):-pertenece(X,[C|Y]),!,ant(X,Y,L).

% Devuelve una lista con los siguientes elementos a X (primera ocurrencia)
% Ejemplo:      sig(4,[1,4,2,4,7],L).      L=[2,4,7]
%              sig(8,[1,4,2,1,7],L).      No

sig(_X,[],):-!.
sig(_X,[_X|_Y],_Y):-!.
sig(X,[C|Y],L):-pertenece(X,[C|Y]),!,sig(X,Y,L).

% Devuelve dos listas, una con los elementos anteriores y otra con los siguientes a un valor X
% Ejemplo:      izq_der_x(3,[1,3,2,6,4],I,D).  I=[1]      D=[2,6,4]
%              izq_der_x(4,[1,3,2,6,4],I,D).  I=[1,3,2,6]  D=[]
%              izq_der_x(1,[1,3,2,6,4],I,D).  I=[]      D=[3,2,6,4]

izq_der_x(X,L,I,D):-ant(X,L,I),sig(X,L,D).

% Invierte la lista en el primer nivel
% Ejemplo:      invertir([3,2,[1,6],7,4],X).    X = [4,7,[1,6],2,3]

invertir([],):-!.
invertir([X],[X]):-!.
invertir([X|M],L):-invertir(M,S),concatenar(S,[X],L).

% Invierte una lista en todos sus niveles
% Ejemplo:      invertirM([3,2,[1,6],7,4],X).    X = [4,7,[6,1],2,3]

invertirM([],):-!.
invertirM([X|M],S):-lista(X),invertirM(X,L),invertirM(M,T),concatenar(T,[L],S).
invertirM([X|M],S):-invertirM(M,T),concatenar(T,[X],S),!.

% Obtener una lista de los elementos mayores que X y otra con los menores que X
% Ejemplo:      xMayMen(3,[2,3,1,7,0],L1,L2).  L1=[2,1,0]  L2=[7]

xMayMen(X,L,Men,May):-xMenores(X,L,Men),xMayores(X,L,May).

% Arma una lista con todas las posiciones de X en L.
% Ejemplo:      ocurrencias(3,[1,3,2,4,65,7,3],L).  L=[2,7]

ocurrencias(X,Y,Z):-invertir(Y,M),secOcurrências(X,M,_R,L),invertir(L,Z),!.

secOcurrências(_X,[_X],1,[1]):-!.
secOcurrências(_X,[_Y],1,[1]):-!.
secOcurrências(X,[X|Xs],R,[R|Zs]):-secOcurrências(X,Xs,R1,Zs),R is R1 + 1.
secOcurrências(X,[_Y|Ys],R,Z):-secOcurrências(X,Ys,R1,Z),R is R1 + 1.

```

```

% Arma una lista con todos los elementos en secuencia creciente a partir del valor de X
% si en el recorrido encuentra un elemento decreciente (aunque sea mayor a X) no lo incluye
% Ejemplo:      xCrece(4,[1,2,3,4,6,5],L).      L=[4,6]
%              xCrece(4,[1,2,3,4,5,6],L).      L=[4,5,6]

```

```

xCrece(_X, [], []).
xCrece(X, [Y|M], [Y|S]) :- X <= Y, !, xCrece(Y, M, S).
xCrece(X, [_Y|M], S) :- xCrece(X, M, S).

```

```

% Arma una lista con todos los elementos en secuencia decreciente hasta el valor de X
% si en el recorrido encuentra un elemento creciente (aunque sea menor a X) no lo incluye
% Ejemplo:      xDecrece(4,[3,2,1,4,5,6],L).      L=[3,2,1]
%              xDecrece(4,[1,2,3,4,5,6],L).      L=[1]

```

```

xDecrece(_X, [], []).
xDecrece(X, [Y|M], [Y|S]) :- X >= Y, !, xDecrece(Y, M, S).
xDecrece(X, [_Y|M], S) :- xDecrece(X, M, S).

```

```

% Devuelve todos los elementos comunes a dos listas
% Ejemplo:      interseccion([1,6,3],[2,1,8,3],L).      L=[1,3]

```

```

interseccion(_L, [], []) :- !.
interseccion([], _L, []) :- !.
interseccion([_X|L], [_X|H], [_X|Z]) :- interseccion(L, H, Z), !.
interseccion([X|L], [R|H], [X|Z]) :- pertenece(X, H), eliminarPri(X, [R|H], L2),
                                     interseccion(L, L2, Z), !.
interseccion([_X|L], [R|H], Z) :- interseccion(L, [R|H], Z), !.

```

## % MATEMATICAS CON LISTAS

% Suma los elementos de la lista (lista con elementos numericos)

% Ejemplo: sumaElem([3,2,2],X). X=7

sumaElem([X],X):-!.

sumaElem([X|Y],S):-sumaElem(Y,T),S is T + X.

% Suma los elementos respectivos de dos listas, generando otra con los resultados.

% Ambas listas deben tener el mismo tamaño.

% Ejemplo: sumaListas([1,7,4],[9,3,6],L). L=[10,10,10]

sumaListas(L1,L2,L):-long(L1,S),long(L2,S),sumaLA(L1,L2,L).

sumaLA([],[],[]):-!.

sumaLA([X|Xs],[Y|Ys],[S|L]):-sumaLA(Xs,Ys,L),S is X + Y.

% Resta los elementos respectivos de dos listas, generando otra con los resultados.

% Ambas listas deben tener el mismo tamaño.

% Ejemplo: restaListas([1,7,4],[9,3,6],L). L=[-8,4,-2]

restaListas(L1,L2,L):-long(L1,S),long(L2,S),restaLA(L1,L2,L).

restaLA([],[],[]):-!.

restaLA([X|Xs],[Y|Ys],[S|L]):-restaLA(Xs,Ys,L),S is X - Y.

% Multiplicación de un número por cada elemento de una lista.

% Ejemplo: multiplicaEscalar(2,[1,2,3,4],L). L=[2,4,6,8]

multiplicaEscalar(\_,[],[]):-!.

multiplicaEscalar(N,[X|L],[Z|Zs]):-multiplicaEscalar(N,L,Zs),Z is X\*N.

% Producto Cartesiano de dos listas.

% Ejemplo: productoCA([1,2,3],[2,3,4],P). P=20.

productoC(L1,L2,S):-long(L1,L),long(L2,L),productoCA(L1,L2,S).

productoCA([],[],0):-!.

productoCA([X|Xs],[Y|Ys],S):-productoCA(Xs,Ys,S2),S3 is X\*Y,S is S2+S3.

% Determina cuantas veces se repite X en una lista.

% Ejemplo: ocurre(1,[3,2,1,6,1],X). X=2

% ocurre(1,[3,2,6],X). X=0

ocurre(\_X,[],0).

ocurre(X,[X|R],N):-ocurre(X,R,N1),N is N1 + 1,!.

ocurre(X,[\_Y|R],N):-ocurre(X,R,N).

% Determina cuantas veces se repite X en una lista (Multinivel)

% Ejemplo: ocurreM(1,[3,2,[1,3,5],1,[6,2,1]],S). S=3

ocurreM(X,L,S):-listaAtomos(L,L1),ocurre(X,L1,S).

% Verifica si M aparece solo una vez en L (primer nivel).

% Ejemplo: unico([1,2],[3,2,[1,2],7]). Yes

% unico(7,[3,2,[1,2],6]). No

unico(M,L):-ocurre(M,L,N),N = 1.

% Verifica si M aparece más de una vez en L (primer nivel).

% Ejemplo: +de1(2,[3,2,[1,2],7]). No

% +de1(2,[3,2,[1,2],6,2]). Yes

+del(M,L):-ocurre(M,L,N),N > 1.

% Calcula la cantidad de elementos iguales que se encuentran en la misma posición en dos listas

% ejemplo: elemIguales([4,5,1],[1,2,3],X). X=0

% elemIguales([1,5,1],[1,2,3],X). X=1

% Si un elemento se repite en otra posición, no se cuenta.

elemIguales(\_L,[],0):-!.

elemIguales([],\_L,0):-!.

elemIguales([\_X],[\_X],1):-!.

```

elemIguales([X|Y],[X|Z],S):-elemIguales(Y,Z,T),!,S is T + 1.
elemIguales([_X|Y],[_R|Z],S):-elemIguales(Y,Z,S).

% Cuenta los elementos que se encuentran en dos listas en diferente posiciones
% Ejemplo:      examinaLista([4,5,1],[1,2,3],X,Y).      X=1      Y=0
%              examinaLista([1,5,1],[1,2,3],X,Y).      X=1      Y=1
% En X devuelve cantidad de elementos iguales en diferente posicion.
% En Y devuelve cantidad de elementos iguales en la misma posicion.

elemIgualesDif([],_L,_N,0).
elemIgualesDif([X|Y],L,P,S):-xElimina(L,P,L1),ocurre(X,L1,S1),P1 is P + 1,
                        elemIgualesDif(Y,L,P1,R),S is S1 + R.
examinaLista(L1,L2,D,S):-elemIgualesDif(L1,L2,1,D),elemIguales(L1,L2,S).

```

## % METODOS DE ORDENAMIENTO

```

% Dada una lista nos dice si esta o no ordenada ascendentemente
% Ejemplo:      ordenada([1,2,3,4,5]).      Yes
%              ordenada([1,2,7,4,5]).      No

ordenada([_X]).
ordenada([X,Y|Ys]):-X<Y,ordenada([Y|Ys]).

% Genera todas las permutaciones de una lista
% Ejemplo:      permutacion([2,3,1],L).      L=[2,3,1]      L=[2,1,3]
%              L=[3,2,1]      L=[3,1,2]
%              L=[1,2,3]      L=[1,3,2]

permutacion([],[]):-!.
permutacion(Xs,[Z|Zs]):-desarma(Z,Xs,Ys),permutacion(Ys,Zs).

desarma(X,[X|Xs],Xs).
desarma(X,[Y|Ys],[Y|Zs]):-desarma(X,Ys,Zs).

% Ordena la lista ascendentemente por Permutacion
% Ejemplo:      ordenaP([1,3,2],L).      L=[1,2,3]
%              ordenaP([1,3,2,6,2],L).      L=[1,2,2,3,6]

ordenaP(X,Y):-permutacion(X,Y),ordenada(Y),!.

% Ordenación Quicksort
% Ejemplo:      quickSort([1,4,3,2,5,7,6],L).      L=[1,2,3,4,5,6,7]

quickSort([],[]).
quickSort([X|Xs],Y):-
xMayMen(X,Xs,Li,Ld),quickSort(Li,LI),quickSort(Ld,LD),concatenar(LI,[X|LD],Y).

% Insercion ordenada de un elemento en una lista ordenada
% Ejemplo:      insertar(3,[1,4,5,7],L).      L=[1,3,4,5,7]

insertar(X,[],[X]).
insertar(X,[Y|Ys],[Y|Zs]):-X>Y,insertar(X,Ys,Zs).
insertar(X,[Y|Ys],[X,Y|Ys]):-X<=Y.

% Ordenacion de lista en forma ascendente
% Ejemplo:      ordenar2([1,2,5,3,3,6,8,7],L).      L=[1,2,3,3,5,6,7,8]

ordenar2([],[]).
ordenar2([X|Xs],Ys):-ordenar2(Xs,Zs),insertar(X,Zs,Ys).

% Ordena una lista en orden descendente
% Ejemplo:      ordenDescen([1,4,3,2],L).      L=[4,3,2,1]

ordenDescen(L,L2):-ordenar2(L,L3),invertir(L3,L2).

```

**% MANIPULACION DE ARBOLES BINARIOS**

% Representacion de arboles binarios:

```
% arbol(nodo,nil,nil)          <= unicamente la raiz.
% arbol(nodo,arbol(hoja,nil,nil),arbol(hoja,nil,nil)) <= raiz y 2 hojas.
%                               a           raiz / nodo
%                               / \
%                               b c           hojas
% Determina si un arbol esta vacio
% Ejemplo: vacioA(arbol(a,nil,arbol(b,nil,nil))).      No
%          vacioA(arbol(nil)).                        Yes
```

vacioA(arbol(nil)):-!.

% Arma una lista con todos los elementos del arbol

```
% Ejemplo: listaArbol(arbol(a,arbol(b,nil,nil),arbol(c,nil,nil)),L).      L=[a,b,c]
```

```
listaArbol(arbol(A,nil,nil),[A]):-!.
listaArbol(arbol(A,X,nil),S):-listaArbol(X,P),concatenar([A],P,S).
listaArbol(arbol(A,nil,X),S):-listaArbol(X,P),concatenar([A],P,S).
listaArbol(arbol(A,X,Y),S):-listaArbol(X,P),listaArbol(Y,R),
                             concatenar([A],P,U),concatenar(U,R,S).
```

% Determina si un elemento pertenece al arbol

```
% Ejemplo: perteneceA(arbol(a,arbol(b,nil,nil),arbol(c,nil,nil)),a).      Yes
%          perteneceA(arbol(a,arbol(b,nil,nil),arbol(c,nil,nil)),z).      No
```

```
perteneceA(arbol(_A,_X,_Y),_A):-!.
perteneceA(arbol(_A,X,_Y),B):-perteneceA(X,B),!.
perteneceA(arbol(_A,_X,Y),B):-perteneceA(Y,B),!.
```

% cuenta la cantidad de elementos que tiene un arbol

```
% Ejemplo: cantElemA(arbol(a,arbol(b,nil,nil),arbol(c,nil,nil)),C).      C=3
```

```
cantElemA(arbol(_A,nil,nil),1):-!.
cantElemA(arbol(_A,X,nil),N):-cantElemA(X,K),N is K + 1.
cantElemA(arbol(_A,nil,X),N):-cantElemA(X,K),N is K + 1.
cantElemA(arbol(_A,X,Y),N):-cantElemA(X,K),cantElemA(Y,T), N is 1 + K + T.
```

% Determina si una arbol binario es completo

```
% Ejemplo: completoA(arbol(a,arbol(b,nil,nil),arbol(c,nil,nil))).      Yes
%          completoA(arbol(a,arbol(b,nil,nil),nil)).                    No
```

```
completoA(arbol(_A,nil,nil)):-!.
completoA(arbol(_A,X,Y)):-profundidadA(X,N),profundidadA(Y,M),N == M,
                             completoA(X),completoA(Y).
```

% Calcula la profundidad del arbol (la raiz tiene nivel 0)

```
% Ejemplo: profundidadA(arbol(a,arbol(b,nil,nil),arbol(c,arbol(e,arbol(f,nil,nil),arbol(j,nil,arbol(z,nil,nil))),nil)),P).
%          P = 4
```

```
profundidadA(arbol(_A,nil,nil),0):-!.
profundidadA(arbol(_A,Y,nil),N):-profundidadA(Y,B),N is B + 1,!.
profundidadA(arbol(_A,nil,X),N):-profundidadA(X,B),N is B + 1,!.
profundidadA(arbol(_A,X,Y),N):-profundidadA(X,B),profundidadA(Y,C), B1 is B+1,
                             C1 is C+1,mayor(B1,C1,N),!.
```

```
mayor(B,C,C):-C>=B,!.
mayor(_B,_C,_B).
```

% Recorre un arbol en preorden

```
% Ejemplo: preordenA(arbol(a,arbol(b,nil,nil),arbol(c,nil,nil)),L).      L=[a,b,c]
```

```
preordenA(arbol(A,nil,nil),[A]):-!.
preordenA(arbol(A,X,nil),[A|S]):-preordenA(X,S),!.
preordenA(arbol(A,nil,X),[A|S]):-preordenA(X,S),!.
preordenA(arbol(A,X,Y),[A|S]):-preordenA(X,T),preordenA(Y,O),concatenar(T,O,S).
```

```

% Recorre un arbol en inorden
% Ejemplo:      inordenA(arbol(a,arbol(b,nil,nil),arbol(c,nil,nil)),L).          L=[b,a,c]

inordenA(arbol(A,nil,nil),[A]) :-!.
inordenA(arbol(A,X,nil),S) :-inordenA(X,C),concatenar(C,[A],S).
inordenA(arbol(A,nil,X),[A|S]) :-inordenA(X,S).
inordenA(arbol(A,X,Y),S) :-inordenA(X,C),inordenA(Y,F),
                               concatenar(C,[A],D),concatenar(D,F,S).

% Recorre un arbol en postorden
% Ejemplo:      postordenA(arbol(a,arbol(b,nil,nil),arbol(c,nil,nil)),L).      L=[b,c,a]

postordenA(arbol(A,nil,nil),[A]) :-!.
postordenA(arbol(A,X,nil),S) :-postordenA(X,C),concatenar(C,[A],S).
postordenA(arbol(A,nil,X),S) :-postordenA(X,C),concatenar(C,[A],S).
postordenA(arbol(A,X,Y),S) :-postordenA(X,C),postordenA(Y,F),
                               concatenar(C,F,D),concatenar(D,[A],S).

% Devuelve el numero de hijos en un arbol binario. Para arboles con solo la raiz, se cuenta 1 hijo (la misma raiz)
% Ejemplo:      cuentaHoja(arbol(a,arbol(b,nil,nil),arbol(e,nil,arbol(k,nil,nil))),X).      X=2
%               cuentaHoja(arbol(a,nil,nil),X).                                         X=1

cuentaHoja(arbol(_R,nil,nil),1) :-!.
cuentaHoja(arbol(_R,Hi,nil),S) :-cuentaHoja(Hi,S).
cuentaHoja(arbol(_R,nil,Hd),S) :-cuentaHoja(Hd,S).
cuentaHoja(arbol(_R,Hi,Hd),S) :-cuentaHoja(Hi,S1),cuentaHoja(Hd,S2),S is S1+S2.

% version 2

cuentaHoja2(nil,0) :-!.
cuentaHoja2(arbol(_R,nil,nil),1) :-!.
cuentaHoja2(arbol(_R,Hi,Hd),S) :-cuentaHoja2(Hi,S1),cuentaHoja2(Hd,S2),S is S1 + S2.

% Suma el valor de todas las hojas (valores numericos)
% Ejemplo:      sumaHoja(arbol(1,nil,arbol(2,arbol(4,nil,nil),arbol(3,nil,nil))),X).      X=7

sumaHoja(nil,0) :-!.
sumaHoja(arbol(R,nil,nil),R) :-!.
sumaHoja(arbol(_R,Hi,Hd),S) :-sumaHoja(Hi,S1),sumaHoja(Hd,S2),S is S1 + S2.

% Extrae un arbol a partir de un nodo X dado.
% Ejemplo:      subArbol(b,arbol(a,arbol(b,arbol(c,nil,nil),arbol(d,nil,nil)),arbol(e,nil,nil)),A).
%               A = arbol(b,arbol(c,nil,nil),arbol(d,nil,nil))
%               subArbol(b,arbol(a,arbol(b,arbol(c,nil,nil),nil),nil),A).
%               A = arbol(b,arbol(c,nil,nil),nil)
%               subArbol(j,arbol(a,arbol(b,nil,nil),arbol(c,arbol(e,arbol(f,nil,nil),arbol(j,nil,arbol(z,nil,nil))),nil)),A).
%               A = arbol(j,nil,arbol(z,nil,nil))

subArbol(X,arbol(X,HI,HD),arbol(X,HI,HD)) :-!.
subArbol(X,arbol(_Y,HI,_HD),R) :-subArbol(X,HI,R).
subArbol(X,arbol(_Y,_HI,HD),R) :-subArbol(X,HD,R).

```

## % MANIPULACION DE GRAFOS

```

% Representacion [[a,b],[b,a],[b,c]] es:
%
%      a <-> b --> c
%
% En los grafos no dirigidos se deben especificar los dos pares de nodos.

% Determina si existe un camino entre X e Y. Devuelve True o False.
% Ejemplo:      camino(c,b,[[a,b],[b,c],[d,e],[c,d],[b,e],[e,c],[e,f],[a,a]]).      No
%               camino(c,f,[[a,b],[b,c],[d,e],[c,d],[b,e],[e,c],[e,f],[a,a]]).      Yes

camino(X,Y,G) :-caminoAux(X,Y,G,[]).
caminoAux(X,Y,G,_T) :-pertenece([X,Y],G).
caminoAux(X,Y,G,T) :-pertenece([X,Z],G),not(pertenece([X,Z],T)),
                       concatenar([X,Z],T,Tt),caminoAux(Z,Y,G,Tt).

```

```

% Devuelve la lista R con los nodos del grafo G.
% Ejemplo:      nodos([[a,b],[b,c],[d,e],[c,d],[b,e],[e,c],[e,f],[a,a]],R).      R=[a, b, c, d, e, f]

nodos(G,R):-listaAtomos(G,L),eliminaR(L,R).

% Cuenta los nodos diferentes del grafo
% Ejemplo:      cuentaNodos([[a,b],[b,c],[d,e],[c,d],[b,e],[e,c],[e,f],[a,a]],R).      R=6

cuentaNodos(G,R):-nodos(G,L),long(L,R).

% Para grafos no dirigidos. Determina si un Grafo es o no conexo.
% Ejemplo:      conexo([[a,b],[b,a],[b,c],[c,b],[c,d],[d,c]]).      Yes
%               conexo([[a,b],[b,a],[b,c],[c,b],[c,d]]).      No

conexo(G):-nodos(G,R),conexoAux(R,G).
conexoAux([],_G):-!.
conexoAux([X|Xs],G):-nodos(G,N),eliminaX(N,X,R),llegaTodos(X,R,G),conexoAux(Xs,G).

llegaTodos(_X,[],_G):-!.
llegaTodos(X,[Y|Ys],G):-camino(X,Y,G),llegaTodos(X,Ys,G).

% Como parámetro se le pasa un camino, un nodo y un grafo.
% Ejemplo:      visita_nodo([a,b,c],b,[[a,b],[b,c],[c,d]]).      Yes
%               visita_nodo([a,b,c],d,[[a,b],[b,c],[c,d]]).      No

visita_nodo(C,N,G):-pertenece(N,C),esCamino(C,G).

% El primer parámetro es una lista indicando un camino. El segundo parámetro
% es el grafo. Responde Yes si existe en el grafo el camino especificado.
% Ejemplo:      esCamino([a,b,c],[[a,b],[b,c],[c,d]]).      Yes
%               esCamino([a,c],[[a,b],[b,c],[c,d]]).      No

esCamino([],_G).
esCamino([X],G):-nodos(G,L),pertenece(X,L).
esCamino([X,Y|Z],G):-pertenece([X,Y],G),esCamino([Y|Z],G).

% Devuelve la longitud de un camino. Se le pasa el inicio y el final.
% Ejemplo:      caminoLongL(a,c,R,[[a,b],[b,c],[c,d]]).      R=2

caminoLongL(X,Y,L,Go):-caminoLongAux(X,Y,L,Go,[]).

caminoLongAux(X,Y,1,Go,_T):-pertenece([X,Y],Go).
caminoLongAux(X,Y,L,Go,T):-pertenece([X,Z],Go),not(pertenece([X,Z],T)),
concatenar([X,Z],T,Ti),caminoLongAux(Z,Y,H,Go,Ti),
L is H + 1.

% Dada una longitud de camino y un grafo, devuelve otro grafo compuesto por pares de nodos
% cuya distancia entre ellos sea la longitud dada en el grafo original (Ejercicio 8, guía 3).
% Ejemplo:      generarGrafo(1,[[a,b],[b,c],[c,d],[b,e]],G).      G=[[a,b],[b,c],[b,e],[c,d]]
%               generarGrafo(2,[[a,b],[b,c],[c,d],[b,e]],G).      G=[[a,c],[a,e],[b,d]]
%               generarGrafo(3,[[a,b],[b,c],[c,d],[b,e]],G).      G=[[a,d]]
%               generarGrafo(4,[[a,b],[b,c],[c,d],[b,e]],G).      G=[]
% Nota: si la longitud es 1, devuelve el grafo original.

generarGrafo(L,Go,Gr):-nodos(Go,R),ggAux(R,R,L,Go,Gr),!.

ggAux([X],R,L,Go,M):-lCaminos(X,R,L,Go,M).
ggAux([X|Y],R,L,Go,B):-lCaminos(X,R,L,Go,A),ggAux(Y,R,L,Go,B1),
concatenar(A,B1,B).

lCaminos(X,[Y],L,Go,[[X,Y]]):-caminoLongL(X,Y,L,Go).
lCaminos(_X,[_Y],_L,_Go,[]).
lCaminos(X,[A|B],L,Go,[[X,A]|N]):-caminoLongL(X,A,L,Go),lCaminos(X,B,L,Go,N),!.
lCaminos(X,[_A|B],L,Go,M):-lCaminos(X,B,L,Go,M).

% Suma los nodos de un grafo. Estos deben ser numeros.
% Ejemplo:      sumaNodos([[1,2],[3,4],[5,6],[2,3],[3,5]],S).      S=21

sumaNodos(G,R):-nodos(G,L),sumaElem(L,R).

```

```
% Determina si existe un rizo en el grafo.
% Ejemplo:      tieneRizo([[a,b],[b,c],[c,d],[a,a]]).      Yes
%              tieneRizo([[a,b],[b,c],[c,d]]).            No
```

```
tieneRizo(G):-nodos(G,L), rizoAux(L,G).
rizoAux([X|_Xs],G):-pertenece([X,X],G).
rizoAux([_X|Xs],G):-rizoAux(Xs,G).
```

```
% 82 - Determina si existe al menos un bucle en el grafo dado.
% Ejemplo:      tieneBucle([[a,b],[b,c],[c,d],[c,b]]).      Yes
%              tieneBucle([[a,b],[b,c],[c,d]]).            No
```

```
tieneBucle([X,Y|G]):-camino(Y,X,G).
tieneBucle([_X,_Y|G]):-tieneBucle(G).
```

```
% Devuelve el camino más corto entre dos nodos.
% Ejemplo:      mejorCamino(a,e,[[a,b],[b,c],[c,d],[b,d],[d,e],[c,e]],C).      C=[a,b,d,e]
%              mejorCamino(a,c,[[a,b],[b,c],[c,d],[b,d],[d,e],[c,e]],C).      C=[a,b,c]
```

```
mejorCamino(I,F,G,C):-camino(I,F,G), mCaminoAux(I,F,G,C), esCamino(C,G), !.
mCaminoAux(I,F,G,[I,F]):-pertenece([I,F],G).
mCaminoAux(I,F,G,[I,C|Cs]):-mCaminoAux(C,F,G,[C|Cs]).
```

### % MANIPULACION DE MATRICES

```
% Formato de matrices: lista de lista.
%              1 2 3
% Ejemplo para N=3 => [[1,2,3], [4,5,6], [7,8,9]]      => 4 5 6
%              7 8 9
% Verifica que el argumento sea una matriz cuadrada.
% Ejemplo:      matrizCuad([[1,2,3],[4,5,6],[7,8,9]]).      Yes
%              matrizCuad([[1,2,3],[4,5,6]]).                No
```

```
matrizCuad(M):-long(M,S), matrizAux(M,S).
matrizAux([],_):-!.
matrizAux([M|Ms],S):-long(M,S), matrizAux(Ms,S).
```

```
% Verifica que el argumento sea una matriz.
% Ejemplo:      esMatriz([[1,2,3],[2,5,4]]).                Yes
%              esMatriz([[1,2,3],[2,5]]).                  No
```

```
esMatriz([F|M]):-long(F,S), esMatrizA(M,S).
esMatrizA([],_):-!.
esMatrizA([F|M],S):-long(F,S), esMatrizA(M,S).
```

```
% Rota una matriz (rotacion antihoraria).
% Ejemplo:      rotarA([[1,2,3],[4,5,6],[7,8,9]],M).        M = [[3,6,9],[2,5,8],[1,4,7]]
%              1 2 3      3 6 9
%              4 5 6 =>  2 5 8
%              7 8 9      1 4 7
```

```
rotarA([],[]):-!.
rotarA(M,[S|M2]):-armarIrenglon(M,S,Ms), rotarA(Ms,M2), !.
rotarA(_,[]).
```

```
armarIrenglon([],[],[]):-!.
armarIrenglon([C|M],[U|R],[S|N]):-ultimoElem(C,U), qu(C,S), armarIrenglon(M,R,N).
```

```

% Rota una matriz (rotacion horaria).
% Ejemplo:      rotarH([[1,2,3],[4,5,6],[7,8,9]],M).           M = [[7,4,1],[8,5,2],[9,6,3]]
%
% 1 2 3          7 4 1
% 4 5 6 =>      8 5 2
% 7 8 9          9 6 3

rotarH([],[]):-!.
rotarH(M,[S|M2]):-armarUNrenglon(M,T,Ms),invertir(T,S),rotarH(Ms,M2),!.
rotarH(_,[]):.

armarUNrenglon([],[],[]):-!.
armarUNrenglon([C|M],[U|R],[S|N]):-primerElem(C,U),qp(C,S),armarUNrenglon(M,R,N).

% Recorrido en espiral de una matriz (sentido horario).
% Ejemplo:      espiralH([[1,2,3],[4,5,6],[7,8,9]],M).         M = [1,2,3,6,9,8,7,4,5]
%
% espiralH([],[]):-!.
% espiralH([E1|M],L):-rotarA(M,R),espiralH(R,L1),concatenar(E1,L1,L).

% Recorrido en espiral de una matriz (sentido antihorario).
% Ejemplo:      espiralA([[1,2,3],[4,5,6],[7,8,9]],M).         M = [1,4,7,8,9,6,3,2,5]
%
% espiralA([],[]):-!.
% espiralA([E1|M],L):-
% rotarH([E1|M],[E2|R]),espiralA(R,L1),invertir(E2,T),concatenar(T,L1,L).

% Igualdad de matrices
% Ejemplo:      igualMatriz([[1,2,3],[4,5,6],[7,8,9]],[[1,2,3],[4,5,6],[7,8,9]]).   Yes
%              igualMatriz([[1,2,3],[4,5,6],[7,8,9]],[[1,2,3],[4,5,6],[7,9,8]]).   No

igualMatriz(M1,M1).

% Suma matrices.
% Ejemplo:      sumaMatrices([[1,2,3],[2,1,0],[0,0,1]],[-1,2,1],[0,0,2],[1,2,7]],S).  S = [[0,4,4],[2,1,2],[1,2,8]]
%              sumaMatrices([[1,2],[2,1],[0,0]],[-1,2],[0,0],[1,2]],S).             S = [[0,4],[2,1],[1,2]]

sumaMatrices([],[],[]):-!.
sumaMatrices([M1|Ms1],[M2|Ms2],[M|Ms]):-sumaMatrices(Ms1,Ms2,Ms),sumaListas(M1,M2,M).

% Resta matrices.
% Ejemplo:      restaMatrices([[1,2,3],[2,1,0],[0,0,1]],[-1,2,1],[0,0,2],[1,2,7]],R).  R = [[2,0,2],[2,1,-2],[-1,-2,-6]]
%              restaMatrices([[1,2],[2,1],[0,0]],[-1,2],[0,0],[1,2]],R).             R = [[2,0],[2,1],[-1,-2]]

restaMatrices([],[],[]):-!.
restaMatrices([M1|Ms1],[M2|Ms2],[M|Ms]):-restaMatrices(Ms1,Ms2,Ms),restaListas(M1,M2,M).

% Producto de matrices.
% Ejemplo:      productoMatricial([[1,2,3],[2,1,0],[0,0,1]],[-1,2,1],[0,0,2],[1,2,7]],M).  M = [[2,8,26],[-2,4,4],[1,2,7]]
%              productoMatricial([[1,2],[2,0],[0,0]],[-1,2,1],[0,0,2]],M).             M = [-1,2,5],[-2,4,2],[0,0,0]]
%              productoMatricial([[3,2,1,-2],[-6,4,0,3]],[[1],[4],[0],[2]],M).         M = [[7],[16]]
%              productoMatricial([-4,5,1],[0,4,2]],[[3,-1,1],[5,6,4],[0,1,2]],M).       M = [[13,35,18],[20,26,20]].

productoMatricial([F|Ms1],M2,M):-long(F,L),long(M2,L),
                                rotarA(M2,M22),productoMA([F|Ms1],M22,M).

productoMA([],_,[]):-!.
productoMA([F|Ms1],M2,[L1|Ls]):-productoMA(Ms1,M2,Ls),pMA(F,M2,L),invertir(L,L1).

pMA(_,[],[]):-!.
pMA(F,[M2|Ms2],[S|L]):-pMA(F,Ms2,L),productoC(F,M2,S).

% Extrae la diagonal principal de una matriz cuadrada.
% Ejemplo:      diagonal([[1,2,3],[4,5,6],[7,8,9]],D).         D=[1,5,9]

diagonal(M,D):-matrizCuad(M),diagonalA(M,1,D1),invertir(D1,D).

diagonalA([],_,[]):-!.
diagonalA([F|M],N,D):-N1 is N + 1,diagonalA(M,N1,D1),nPosicion(F,N,E),concatenar(D1,E,D).

```

```

% Matriz escalonada inferior en matrices cuadradas.
% Ejemplo:      escalonadaI([[1,0,0],[4,5,0],[7,8,9]]).      Yes
%              escalonadaI([[1,0,1],[4,5,0],[7,8,9]]).      No
%
% 1 0 0          1 0 1          Un matriz se dice escalonada cuando al pasar de un renglón a otro,
% 4 5 0 => Yes   4 5 0 => No     la cantidad de 0 (ceros) se va incrementando.
% 7 8 9          7 8 9

```

```

escalonadaI(M):-matrizCuad(M),escalonadaIA(M,1,R),sumaElem(R,S),S=0.
escalonadaIA([],_,[]):-!.
escalonadaIA([F|M],P,L):-P1 is P+1,escalonadaIA(M,P1,L1),
                          sacaNpri(F,P,R),concatenar(R,L1,L).

```

```

% Matriz escalonada superior en matrices cuadradas.
% Ejemplo:      escalonadaS([[1,0,0],[4,5,0],[7,8,9]]).      Yes
%              escalonadaS([[1,0,4],[4,5,0],[7,8,9]]).      No
%
% 1 0 0          1 0 4
% 4 5 0 => Yes   4 5 0 => No
% 7 8 9          7 8 9

```

```

escalonadaS(M):-rotarH(M,M1),rotarH(M1,M2),escalonadaI(M2).

```

```

% Verifica si una matriz es la Matriz Identidad (matrices cuadradas).
% Ejemplo:      identidad?([[1,0,0],[0,1,0],[0,0,1]]).      Yes
%              identidad?([[1,0,4],[4,5,0],[7,8,9]]).      No
%              identidad?([[1,0,0],[0,1,0],[0,1,1]]).      No

```

```

identidad?([F|M]):-
escalonadaI([F|M]),escalonadaS([F|M]),long(F,R),diagonal([F|M],L),sumaElem(L,R).

```

```

% Multiplicar una matriz por un numero escalar.
% Ejemplo:      multMatrizEscalar(2,[[1,1,1],[1,1,1],[1,1,1]],M).      M = [[2,2,2],[2,2,2],[2,2,2]]
%              multMatrizEscalar(2,[[4,5,6],[1,2,3]],M).              M = [[8,10,12],[2,4,6]]

```

```

multMatrizEscalar(_,[],[]):-!.
multMatrizEscalar(X,[F|M],[L|Ls]):-multMatrizEscalar(X,M,Ls),multiplicaEscalar(X,F,L).

```

```

% Devuelve la diagonal n-esima de una matriz.
% Ejemplo:      diagonalN([[1,2,3],[4,5,6],[7,8,9]],1,R).      R = [1,5,9]
%              diagonalN([[1,2,3],[4,5,6],[7,8,9]],2,R).      R = [2,6]

```

```

diagonalN([],_,[]):-!.
diagonalN([M|Ms],N,[]):-long(M,L),L < N,!.
diagonalN([M|Ms],N,D):-N1 is N+1,diagonalN(Ms,N1,D1),
                          nPosicion(M,N,E),concatenar(E,D1,D).

```

```

% Devuelve la matriz original transpuesta ((i,j)=>[j,i]).
% Ejemplo:      transpuesta([[1,2,3],[4,5,6],[7,8,9]],M).      M = [[1,4,7],[2,5,8],[3,6,9]]

```

```

transpuesta(M,Mt):-rotarH(M,Mr),transpuestaAux(Mr,Mt).
transpuestaAux([X],[Y]):-invertir(X,Y),!.
transpuestaAux([X|Xs],[Y|Ys]):-invertir(X,Y),transpuestaAux(Xs,Ys).

```

```

% Dimensiones en matrices: matrizFil retorna las filas de una matriz,
% y matrizCol las columnas.

```

```

matrizFil(M,F):-long(M,F).
matrizCol([M|Ms],C):-long(M,C).

```

```

% Dada una matriz, busca el elemento en las coordenadas [x,y] dadas.
% Ejemplo:      matrizXY([[1,2,3],[4,5,6],[7,8,9]],2,3,V).      V=8

```

```

matrizXY(M,C,F,V):-matrizFil(M,F1),matrizCol(M,C1),F=<F1,C=<C1,
                  enesimoElem(M,F,A),enesimoElem(A,C,V).

```