

% EJERCICIOS DE EXAMEN

% 2003

% Devuelve todas las sublistas de S cuya suma de elementos es igual a E

```
% Ejemplo:      subSecR([1,2,3,4,5,6],3,L).      L=[1,2]      L=[3]
%              subSecR([1,2,3,4,5,6],7,L).      L=[3,4]
```

```
subSecR(S,E,R):-subSec(S,R),sumaElem(R,E).
```

% Ejercicio de examen. 28/02/2005

```
% Verifica si la suma de los elementos anteriores a aquel cuya posicion
% se especifica es igual al elemento cuya posicion se especifica.
```

```
% Ejemplo:      sumaAnterior([1,7,8,16,5],4).      Yes
%              sumaAnterior([1,7,7,16,5],4).      No      (1 + 7 + 7 = 15 <> 16)
```

```
sumaAnterior(L,N):-nPrimerosT(L,L1),qu(L1,L2),sumaElem(L2,S),
enesimoElem(L,N,S).
```

% Diciembre-2004

% asignar(Lnum,Grafo,Restricciones,Lasignaciones).

% Lasignaciones es un predicado no-ground

% asignar([1,2,3,4,5,6,7,8,9],

```
%      [[a,b],[b,c],[c,a],[a,e],[e,f],[f,c],[c,d],[b,d],[d,g],[g,d],[g,h],[h,f],[h,i],[f,i],[e,i]],
```

```
%      [[a,b,c,12],[b,c,d,11],[a,c,e,f,28],[c,d,f,g,20],[f,f,e,22],[f,f,h,16],[f,g,h,13]],La).
```

%

% Dado un grafo, el valor de las restricciones surge de sumar los valores

% de los nodos que se vinculan. "La" devuelve una lista de nodos con los

% valores que deben tener para satisfacer los criterios.

%

```
%      La=[[a,7],[b,1],[c,4],[e,9],[f,8],[d,6],[g,2],[h,...], [...]]
```

%

% version 1

```
asignar1(N,G,R,A):-nodos(G,No),permutacion(N,Np),
combinar1(No,Np,A),contemplar1(R,A),!.
```

```
contemplar1([],_).
```

```
contemplar1([R|Rs],A):-contemplarAux1(R,A),contemplar1(Rs,A).
```

```
contemplarAux1(R,A):-pR1(R,A,L),sumaElem(L,M),invertir(R,[H|_Hs]),H=M.
```

```
pR1(_R,[],[]).
```

```
pR1(R,[[X,_L]|As],[_L|Ls]):-pertenece(X,R),pR1(R,As,Ls).
```

```
pR1(R,[_A|As],L):-pR1(R,As,L).
```

```
combinar1([],[],[]).
```

```
combinar1([_N|Ns],[_M|Ms],[[_N,_M]|S]):-combinar1(Ns,Ms,S).
```

% version 2

```
asignar2(N,G,R,A):-nodos(G,No),permutacion(N,Np),combinar1(No,Np,A),validar2(R,A),!.
```

```
validar2([],_A).
```

```
validar2([R|Rs],A):-nUltimos(R,1,U),long(R,L),L1 is L-1,nPrimeros(R,L1,N),
validarAux2(N,U,A),validar2(Rs,A).
```

```
validarAux2(N,[S],A):-suma2(N,A,S).
```

```
suma2([E],A,S):-pertenece([E,S],A).
```

```
suma2([E|Es],A,S):-pertenece([E,S1],A),suma2(Es,A,S2),S is S1 +S2.
```

% TORTUGA (ninja ;) N=3

% 24/02/03

% Debe proporcionarse el tamaño del tablero, las coordenadas (X,Y) de las celdas
 % y el número de pasos en que deben ser alcanzadas por la tortuga. En C se
 % informa los caminos que se encuentren que satisfagan la restricción (celda en
 % en tantos pasos. Se asume que la tortuga no necesita recorrer todas las celdas.

%

% Ejemplo: tortuga1(4,[[2,1,1],[3,3,6]],C).

%

% 1 de tantos caminos: C=[[2,1,1],[1,1,2],[1,2,3],[2,2,4],[3,2,5],[3,3,6]]

% version 1

```
tortuga1(N,M,C) :- celdaC(X,Y), moverC([X,Y,1],[X1,Y1,2],N),
                  tAux1(N,M,[X,Y,1],[X1,Y1,2],C).
```

```
tAux1(_N,M,C,C) :- pertenece([1,1,_],C), verificaC(C,M), !.
```

```
tAux1(N,M,C1,C) :- ultimoElem(C1,Cu), moverC(Cu,[X,Y,S],N),
                  not(pertenece([X,Y,_],C1)), concatenarElem(C1,[X,Y,S],Cn),
                  tAux1(N,M,Cn,C).
```

% version 2

```
tortuga2(N,M,C) :- caminos2(N,C), verificaC(C,M).
```

```
caminos2(N,[X,Y,1|Z]) :- celdaC(X,Y), caminoAux2(N,[X,Y,1|Z],[]).
```

```
caminoAux2(_N,[1,1,_],_) :- !.
```

```
caminoAux2(N,[X,Y,Z],[X1,Y1,Z1|M],A) :- moverC([X,Y,Z],[X1,Y1,Z1],N),
                                         not(pertenece([X1,Y1,_],A)),
```

```
concatenar([X,Y,Z],A,A1), caminoAux2(N,[X1,Y1,Z1|M],A1).
```

% Funciones Auxiliares

```
moverC([X,Y,N],[X,Y1,N1],T) :- N1 is N+1, Y1 is Y+1, Y1 < T.
```

```
moverC([X,Y,N],[X1,Y,N1],T) :- N1 is N+1, X1 is X+1, X1 < T.
```

```
moverC([X,Y,N],[X,Y1,N1],_T) :- N1 is N+1, Y1 is Y-1, Y1 > 0.
```

```
moverC([X,Y,N],[X1,Y,N1],_T) :- N1 is N+1, X1 is X-1, X1 > 0.
```

```
verificaC(_C,[]) :- !.
```

```
verificaC(C,[M|Ms]) :- pertenece(M,C), verificaC(C,Ms).
```

% Tablero

```
celdaC(1,1).
```

```
celdaC(1,2).
```

```
celdaC(1,3).
```

```
celdaC(2,1).
```

```
celdaC(2,2).
```

```
celdaC(2,3).
```

```
celdaC(3,1).
```

```
celdaC(3,2).
```

```
celdaC(3,3).
```

% TAREAS- Examen Final del 28/07/03

% Debe armarse una lista de tareas (que deben hacerse en un cierto orden) que
 % represente el menor costo.

% Datos

```
tarea(nombre(nil),precede(t1)).
```

```
tarea(nombre(t1),precede(t2)).
```

```
tarea(nombre(t1),precede(t3)).
```

```
tarea(nombre(t2),precede(t4)).
```

```
tarea(nombre(t3),precede(t4)).
```

```
tarea(nombre(t4),precede(nil)).
```

```

recursoTarea(t1,100,r1).
recursoTarea(t2,10,r1).
recursoTarea(t3,5,r2).
recursoTarea(t4,120,r1).

recurso(r1,'informatico').
recurso(r2,'humano').

% Predicados
% Ejemplo:      secuenciaTareas(L).
%              L=[nodoS(t1,informatico,100),nodoS(t3,humano,5),nodoS(t4,informatico,120)]

secuenciaTareas(L):-armarLdeL3(M,[ ]),listaMenorCosto(M,L).

armarLdeL3([X|L],Aux):-armarLista3(X,nil),not(pertenece(X,Aux)),!,
                    concatenar(Aux,[X],Aux2),armarLdeL3(L,Aux2).
armarLdeL3([ ],_Aux).

armarLista3([ ],E):-tarea(nombre(E),precede(nil)).
armarLista3([nodoS(T,D,C)|S],E):-tarea(nombre(E),precede(T)),
                                recursoTarea(T,C,I),recurso(I,D),
                                armarLista3(S,T).

listaMenorCosto([M|Ms],R):-sumaCosto(M,Cr),menorCostoAux(Ms,R,Cr,M),!.

menorCostoAux([ ],_M,_Cr,_M):-!.
menorCostoAux([M|Ms],R,Cr,_N):-sumaCosto(M,C),C<Cr,menorCostoAux(Ms,R,C,M).
menorCostoAux([_M|Ms],R,Cr,N):-menorCostoAux(Ms,R,Cr,N).

sumaCosto([nodoS(_,_),C],C).
sumaCosto([nodoS(_,_),C]|S],C1):-sumaCosto(S,C2),C1 is C + C2.

% JULIO CESAR - Examen final del 29/07/02
%
% Si hay piratas en el camino, se resta 1 tripulante y se suman 2 en cada isla alcanzada.
%
% ?- julioCesar(L,30,X).
% L = [15, 16, 12, 11, 17, 14, 13].
% X = 39
%
% L = [15, 16, 14, 13].
% X = 33
%
% L = [15, 16, 17, 14, 13].
% X = 36

% DATOS %

isla(11).
isla(12).
isla(13).
isla(14).
isla(15).
isla(16).
isla(17).

% ruta(extremo1,extremo2,nroPiratas)

ruta(11,12,32).
ruta(12,16,0).
ruta(11,17,0).
ruta(15,16,0).
ruta(13,14,0).
ruta(17,14,0).
ruta(16,14,25).
ruta(16,17,0).

```

```

% camino(extremo1,extremo2,nroPiratas) los caminos son bidireccionales,
% el camino 11-12 es el mismo que el 12-11.

caminoJC(C1,C2,P):-ruta(C1,C2,P).
caminoJC(C1,C2,P):-ruta(C2,C1,P).

% Ubicación de la Sirena

sirena(13).

% Ubicación del Campitan

capitan(15).

% PROGRAMA %

% julioCesar(islas,tripInic,tripFinal)

julioCesar(L,N,X1):-capitan(C),julioAuxJC(L,N,X,C,[]),X1 is X-2.

% julioAux(islas,tripInic,tripFinal,capitan,visitadas)

julioAuxJC([L1,L2],N,X,L1,_T):-puedeAvanzarJC(L1,L2,N,X),sirena(L2).
julioAuxJC([L1|Ls],N,X,L1,T):-isla(L2),not(pertenece(L2,T)),
    puedeAvanzarJC(L1,L2,N,X1),
    concatenar([L2],T,Ti),julioAuxJC(Ls,X1,X,L2,Ti).

% puedeAvanzar(origen,destino,tripInic,tripFinal)

puedeAvanzarJC(L1,L2,N,X):-caminoJC(L1,L2,0),X is N+2,!.
puedeAvanzarJC(L1,L2,N,X):-caminoJC(L1,L2,P),P < N,X is N + 1.

% CAZADOR - Examen final del 03/05/05

% Para llegar a su presa, el cazador debe desplazarse diagonalmente por el tablero.
% Se especifica el tamaño del tablero (N), posición del Cazador (C) y de la presa (P).
%
% Ejemplo:      cazador(3,[1,2],[1,2]).                Yes
%
% Nota:
% Si la respuesta es NO, se sale fuera de la pila antes de responder.
% Si necesita más pasos también se sale de la pila. ERROR: Out of global stack

cazador(N,C,P):-arribaDerecha(N,C,C1),abajoDerecha(N,C1,C2),
    abajoIzquierda(N,C2,C3),arribaIzquierda(N,C3,P).

arribaDerecha(N,[X,Y],[X1,Y1]):-X1 is X + 1, Y1 is Y+1, X1 = <N, Y1 = <N.
arribaDerecha(N,[X,Y],[X2,Y2]):-
arribaDerecha(N,[X,Y],[X1,Y1]),arribaDerecha(X,[X1,Y1],[X2,Y2]).

abajoDerecha(N,[X,Y],[X1,Y1]):-X1 is X + 1, Y1 is Y-1, X1 = <N, Y1 >= 1.
abajoDerecha(N,[X,Y],[X2,Y2]):-
abajoDerecha(N,[X,Y],[X1,Y1]),abajoDerecha(X,[X1,Y1],[X2,Y2]).

abajoIzquierda(N,[X,Y],[X1,Y1]):-X1 is X - 1, Y1 is Y-1, X1 >= 1, Y1 >= 1.
abajoIzquierda(N,[X,Y],[X2,Y2]):-
abajoIzquierda(N,[X,Y],[X1,Y1]),abajoIzquierda(X,[X1,Y1],[X2,Y2]).

arribaIzquierda(N,[X,Y],[X1,Y1]):-X1 is X - 1, Y1 is Y+1, X1 >= 1, Y1 = <N.
arribaIzquierda(N,[X,Y],[X2,Y2]):-
arribaIzquierda(N,[X,Y],[X1,Y1]),arribaIzquierda(X,[X1,Y1],[X2,Y2]).

```

% ARMAR LA DIETA - Examen 05/08/2002

```
%
% Se trata de armar la dieta por dia segun la cantidad de horas de estudio.
% Se informa una lista de lista cuyos elementos son [dia,horas], se
% devuelven las posibles dietas que pueden hacerse sumando las calorías y de
% acuerdo a las restricciones de calorías por horas de estudio (horasCalorias).
%
% Ejemplo:      armarDieta([[lunes,2],[martes,4],[miercoles,3],[jueves,2],[viernes,3]],D).
% una de tantas soluciones:
%              D=[[mantecaSancor],[pepsi],[mantecaSancor],[mantecaSancor],[mantecaSancor]]
```

```
producto(manteca,mantecaSancor,150).
producto(agua,nestle,60).
producto(manteca,mantecaLaSerenisima,200).
producto(leche,lecheSindor,190).
producto(margarina,danica,200).
producto(gaseosa,pepsi,300).
producto(leche,lecheLaCabania,102).
producto(leche,lecheCotapa,90).
producto(carne,vaca,200).
producto(mayonesa,hellman,190).
producto(huevos,cordoniz,140).
producto(gaseosa,frescor,140).
producto(pan,centeno,100).
producto(pan,lactal,190).
producto(pan,harina,120).
producto(leche,lechePolvo,110).
producto(arroz,arrozSanSalvador,110).
producto(mayonesa,mayoDanica,180).
producto(manteca,fredo,170).
producto(agua,aguaSanSalvador,70).
producto(gaseosa,cocaCola,120).
producto(gaseosa,sevenUp,150).
```

```
horasCalorias(1,50).
horasCalorias(2,100).
horasCalorias(3,100).
horasCalorias(4,200).
horasCalorias(5,200).
horasCalorias(6,300).
horasCalorias(7,300).
horasCalorias(8,400).
```

```
armarDieta([],[]).
armarDieta([[_X1,X2]|Xs],[Y|Ys]):-horasCalorias(X2,D),dieta(Y,[],0,D),
                                armarDieta(Xs,Ys).
```

```
dieta([Y],AUX,CAL,CM):-producto(T,Y,C),
                        ocurre(producto(T,_,_),AUX,R),R<4,
                        ocurre(producto(T,Y,C),AUX,M),M<3, N is C + CAL,CM < N.
```

```
dieta([Y|Yr],AUX,CAL,CM):-producto(T,Y,C),
                        ocurre(producto(T,_,_),AUX,R),R<4,
                        ocurre(producto(T,Y,C),AUX,M),M<3,
                        concatenar(AUX,[producto(T,Y,C)],AUX2),N is C + CAL,
                        CM > N, dieta(Yr,AUX2,N,CM).
```

% MUJER FIEL - Examen final

```
%
% Dado una serie de predicados, se trata de averiguar si una mujer ha sido fiel.
```

```
pareja(benito, andrea, [03,10,2001],[10,08,2002]).
pareja(carlos, maria-jose, [01,05,2001],[02,05,2001]).
pareja(santiago, andrea, [02,02,1998],[10,08,1998]).
pareja(santiago, stella, [06,03,1998],[12,06,1998]).
pareja(juan, maria-jose, [03,07,2000],[10,08,2002]).
pareja(pepe, andrea, [03,10,2001],[10,08,2002]).
```

```
% formato [01,02,1998]
```

```
fma([_A,_B,C],[_X,_Y,Z]):-C>Z,!.
fma([_A,B,C],[_X,Y,Z]) :-C==Z,B>Y,!.
fma([A,B,C],[X,Y,Z]) :-C==Z,B==Y,A>=X,!.

```

```
% el segundo par de fechas es posterior al primero
```

```
superpuestas(A,B,C,_D):-fma(C,A),fma(B,C).
```

```
mujerInfidel(M):-pareja(V1,M,Fi1,Ff1),pareja(V2,M,Fi2,Ff2),not(V1=V2),
superpuestas(Fi1,Ff1,Fi2,Ff2).
```

```
mujerFiel(M):-not(mujerInfidel(M)).
```

% EJERCICIO DE EXAMEN - JUEGO 4 EN LÍNEA

```
%jugada(Tablero,ColorFicha,NroColumna,TableroResultado).
```

```
jugada(T,F,N,T):-rotarH(T,Tr),enesimoElem(Tr,N,C),not(pertenece(x,C)),!.
jugada(T,F,N,R):-rotarH(T,Tr),enesimoElem(Tr,N,C),xPosicion(x,C,P),
insertarXenN(C,P,F,C1),insertarXenN(Tr,N,C1,Tra),rotarA(Tra,R).
```

```

%?- jugada([[v,x,v,v,v,n],
%          [v,x,v,v,v,n],
%          [v,v,v,v,v,n],
%          [v,v,v,v,v,n],
%          [v,v,v,v,v,n],
%          [v,v,v,v,v,n],
%          [v,v,v,v,v,n]],n,2,R).
%R = [[v,x,v,v,v,n],
%     [v,n,v,v,v,n],
%     [v,v,v,v,v,n],
%     [v,v,v,v,v,n],
%     [v,v,v,v,v,n],
%     [v,v,v,v,v,n],
%     [v,v,v,v,v,n]]

```

```
% cuatroEnLinea(Tablero,Color). Responde yes si hay cuatro
% fichas en linea de color Color.
```

```
cuatroEnLinea(T,C):-checkHorizontal(T,C),!.
cuatroEnLinea(T,C):-checkVertical(T,C),!.
cuatroEnLinea(T,C):-checkDiagonal(T,C),!.
```

```
checkHorizontal([F|Fs],C):-subLista([C,C,C,C],F),!.
checkHorizontal([F|Fs],C):-checkHorizontal(Fs,C).
```

```
checkVertical(T,C):-rotarH(T,Tr),checkHorizontal(Tr,C).
```

```
checkDiagonal(T,C):-obtenerDiagonal(T,D),subLista([C,C,C,C],D).
```

```
obtenerDiagonal(M,D):-aux41(N),diagonalN(M,N,D).
obtenerDiagonal(M,D):-rotarA(M,Mr),aux41(N),diagonalN(Mr,N,D).
obtenerDiagonal(M,D):-rotarA(M,Mr1),rotarA(Mr1,Mr2),aux41(N),diagonalN(Mr2,N,D).
obtenerDiagonal(M,D):-rotarH(M,Mr),aux41(N),diagonalN(Mr,N,D).
```

```
aux41(1).
aux41(2).
aux41(3).
```

```
%Ejemplos.. aceptamos que las piezas floten :)
```

```
% cuatroEnLinea([[x,v,x,x,x],[x,x,v,x,x],[x,x,x,v,x],[x,x,x,x,v],[x,x,x,x,x],[x,v,v,x,v,x],[x,n,n,v,n,x]],v).
%
% Yes
% cuatroEnLinea([[x,v,x,x,x],[x,x,v,x,n],[x,x,x,v,n,x],[x,x,n,v,x],[x,x,n,x,x],[x,v,v,x,v,x],[x,n,n,v,n,x]],n).
% Yes

```