/\*1-Determina si lo que recibe es una lista\*/

lista([]):-!.

lista([X|Y]):-lista(Y).

/\*------------------------------------------------------------------\*/

/\*2-Concatena dos listas\*/

concatenar([],L,L).

concatenar([X|M],L,[X|Z]):-concatenar(M,L,Z).

/\*------------------------------------------------------------------\*/

/\*3-Invierte la lista que recibe en el primer nivel\*/

invertir([X],[X]).

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

/\*------------------------------------------------------------------\*/

/\*4-Calcula la longitud de la lista\*/

long([],0):-!.

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

/\*------------------------------------------------------------------\*/

/\*5-Suma los elementos de la lista\*/

suma\_elem([X],X):-!.

suma\_elem([X|Y],S):-suma\_elem(Y,T), S is T + X.

/\*------------------------------------------------------------------\*/

/\*6-Devuelve el elemento que se encuentra en la enesima posicion\*/

posicion\_n([],N,[]):-!.

posicion\_n([X|N],1,X):-!.

posicion\_n([X|R],N,S):-M is N - 1,posicion\_n(R,M,S).

/\*------------------------------------------------------------------\*/

/\*7-Dan como resultado el primer y el ultimo elemento de la lista\*/

primer\_elem([X|Y],X):-!.

ultimo\_elem(L,S):-invertir(L,T),primer\_elem(T,S).

/\*------------------------------------------------------------------\*/

/\*8-Calcula la primer posicion donde aparece el elemento X en la lista\*/

prim\_pos(X,[],0):-!.

prim\_pos(X,[X|M],1):-!.

prim\_pos(X,[Y|M],S):-prim\_pos(X,M,T),S is T + 1.

/\*------------------------------------------------------------------\*/

/\*9-Elimina el elemento x de la lista en todos los niveles\*/

elimina\_x([],X,[]):-!.

elimina\_x([X],X,[]):-!.

elimina\_x([X|M],X,S):-elimina\_x(M,X,S),!.

elimina\_x([R|M],X,S):-lista(R), elimina\_x(R,X,T), elimina\_x(M,X,P), concatenar([T],P,S).

elimina\_x([R|M],X,S):-elimina\_x(M,X,T), concatenar([R],T,S).

/\*------------------------------------------------------------------\*/

/\*10-Invierte una lista en todos sus niveles\*/

invertir\_tot([],[]):-!.

invertir\_tot([X|M],S):-lista(X),invertir\_tot(X,P),invertir\_tot(M,T),concatenar(T,[P],S).

invertir\_tot([X|M],S):-invertir\_tot(M,T),concatenar(T,[X],S),!.

/\*------------------------------------------------------------------\*/

/\*11-Saca los n primeros elementos de una lista y devuelve el resto\*/

sac\_n\_pri([],N,[]):-!.

sac\_n\_pri([X|M],1,M):-!.

sac\_n\_pri([X|M],N,S):-N1 is N-1,sac\_n\_pri(M,N1,S).

/\*------------------------------------------------------------------\*/

/\*12-Da como resultado los n primeros elementos de una lista\*/

da\_n\_pri(L,0,[]):-!.

da\_n\_pri([],N,[]):-!.

da\_n\_pri([X|M],1,[X]):-!.

da\_n\_pri([X|M],N,S):-N1 is N - 1,da\_n\_pri(M,N1,T),concatenar([X],T,S).

/\*------------------------------------------------------------------\*/

/\*13-Muestra los ultimos n elementos de la lista\*/

da\_n\_ultim(L,N,S):-invertir\_tot(L,T),da\_n\_pri(T,N,R),invertir\_tot(R,S).

/\*------------------------------------------------------------------\*/

/\*14-Determina si dos listas son iguales en todos los niveles\*/

lista\_igual([],[]):-!.

lista\_igual([X|M],[X|R]):-lista\_igual(M,R).

lista\_igual([X|M],[T|R]):-lista(X), lista(T), lista\_igual(X,T),lista\_igual(M,R).

/\*------------------------------------------------------------------\*/

/\*15-Determina si la primer lista es prefijo de la segunda\*/

prefijo([],M):-!.

prefijo([X],[X|M]):-!.

prefijo([X|L],[X|M]):-prefijo(L,M).

prefijo([X|T],[L|M]):-lista(X),prefijo(X,L),prefijo(T,M).

/\*------------------------------------------------------------------\*/

/\*16-Determina si la primer lista es sublista de la segunda\*/

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).

/\*------------------------------------------------------------------\*/

/\*17-Elimina todos los elementos de la lista 1 que est n en la 2\*/

elim\_l1\_de\_l2([],L,L):-!.

elim\_l1\_de\_l2([X|M],L,S):-elimina\_x(L,X,T),elim\_l1\_de\_l2(M,T,S).

/\*------------------------------------------------------------------\*/

/\*18-Elimina los elementos repetidos que est n en una lista\*/

elim\_repet([],[]):-!.

elim\_repet([X|M],S):-not lista(X), elimina\_x(M,X,T),elim\_repet(T,Y),

concatenar([X],Y,S).

elim\_repet([X|M],S):-lista(X), elim\_l1\_de\_l2(X,M,T),elim\_repet(X,Y),

elim\_repet(T,J),concatenar([Y],J,S).

/\*------------------------------------------------------------------\*/

/\*19-Arma una lista con todos los elementos que se encuentran antes \*/

/\*de elemento x en la lista\*/

hasta\_x([X|M],X,[]):-!.

hasta\_x([Y|M],X,[Y|S]):-hasta\_x(M,X,S).

/\*------------------------------------------------------------------\*/

/\*20-Arma una lista con todos los elementos que se encuentran despues\*/

/\*del elemento x\*/

desde\_x([Y|M],X,S):-X=\=Y,desde\_x(M,X,S),!.

desde\_x([X,Y|M],X,[Y|M]):-X=\=Y,!.

desde\_x([Y|M],X,S):-desde\_x(M,X,S),!.

/\*------------------------------------------------------------------\*/

/\*21-Determina si una lista se encuentra en orden creciente o decreciente\*/

creciente([X,Y|Z]):- X<Y,

creciente([Y|Z]).

creciente([X]).

decreciente([X,Y|Z]):- X>Y,

decreciente([Y|Z]).

decreciente([X]).

/\*------------------------------------------------------------------\*/

/\*22-Cuenta la ocurrencia de una lista at¢mica en otra lista\*/

secuencia([],L,L):-!.

secuencia([X|Y],[X|L],[X|L]):-secuencia(Y,L,L).

cta\_ocurr(L,[],0):-!.

cta\_ocurr([X|Y],[X|M],N):-secuencia(Y,M,L1), cta\_ocurr([X|Y],L1,T), N is T + 1.

cta\_ocurr([X|Y],[R|M],N):-cta\_ocurr([X|Y],M,N),!.

/\*------------------------------------------------------------------\*/

/\*23-Determina si una lista puede analizarse del derecho o del rev‚s\*/

reversible([]):-!.

reversible(L):-invertir(L,R),lista\_igual(L,R).

/\*------------------------------------------------------------------\*/

/\*24-Calcula la cantidad de elementos que se encuentran en la misma\*/

/\*posicion en dos listas\*/

orden(L,[],0):-!.

orden([],L,0):-!.

orden([X],[X],1):-!.

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

orden([X|Y],[R|Z],S):-orden(Y,Z,S).

/\*------------------------------------------------------------------\*/

/\*25-Arma una lista con todas las posiciones del elemento X en la lista\*/

list\_posic(X,[],N,[]):-!.

list\_posic(X,[X|M],N,L):-N1 is N + 1,list\_posic(X,M,N1,G),P is N + 1,

concatenar([P],G,L).

list\_posic(X,[Y|M],N,L):-N1 is N + 1,list\_posic(X,M,N1,L).

concatenar([],L,L):-!.

concatenar([X|Y],L,[X|Z]):-concatenar(Y,L,Z).

/\*------------------------------------------------------------------\*/

/\*26-Arma una lista con la primera posicion de cada elemento de l1 en l2\*/

posic\_l1\_l2([],L,[]):-!.

posic\_l1\_l2([X|Y],L,F):-posic\_l1\_l2(Y,L,L2),prim\_pos(X,L,S),concatenar([S],L2,F).

/\*------------------------------------------------------------------\*/

/\*27-Arma una lista con todos los elementos en secuencia creciente a partir \*/

/\*de X\*/

mayores(X,[],[]).

mayores(X,[Y|M],[Y|S]):-X=<Y,mayores(Y,M,S).

mayores(X,[Y|M],S):-mayores(X,M,S).

/\*------------------------------------------------------------------\*/

/\*28-Arma una lista en orden decreciente a partir del elemento X\*/

menores(X,[],[]).

menores(X,[Y|Z],[Y|R]):-X>=Y,menores(Y,Z,R).

menores(X,[Y|Z],R):-menores(X,Z,R).

/\*------------------------------------------------------------------\*/

/\*29-Arma una lista con todos los elementos menores que el elemento X\*/

menores\_x(X,[],[]):-!.

menores\_x(X,[Y|W],[Y|Z]):-X>Y,menores\_x(X,W,Z),!.

menores\_x(X,[Y|W],Z):-menores\_x(X,W,Z),!.

/\*------------------------------------------------------------------\*/

/\*30-Arma una lista con todos los elementos mayores que el elemento X\*/

mayores\_x(X,[],[]):-!.

mayores\_x(X,[Y|W],[Y|Z]):-X<Y,mayores\_x(X,W,Z),!.

mayores\_x(X,[Y|W],Z):-mayores\_x(X,W,Z),!.

/\*------------------------------------------------------------------\*/

/\*31-Obtener una lista de los elementos menores que X y otra con los \*/

/\* elementos mayores que X\*/

obtenerlista(X,L,Men,May):-menores\_x(X,L,Men),mayores\_x(X,L,May).

/\*------------------------------------------------------------------\*/

/\*32-Arma una lista con las posiciones de la subsecuencia creciente en la\*/

/\*lista de longitud n\*/

long([],0):-!.

long([X|M],S):-long(M,T), S is T+1.

subsec([X|Y],N,L):-mayores(X,[X|Y],S),long(S,N),posic\_l1\_l2(S,[X|Y],L).

subsec([X|Y],N,L):-subsec(Y,N,L).

/\*------------------------------------------------------------------\*/

/\*33-Calcula el cociente y el resto entre dos números positivos\*/

cociente(0,N,0,0):-!.

cociente(N1,N,C,R):-C is N1//N,T is C\*N,R is N1-T.

/\*------------------------------------------------------------------\*/

/\*34-Transforma un número decimal a binario en forma de lista\*/

dec\_bin(1,[1]):-!.

dec\_bin(0,[]):-!.

dec\_bin(N,S):-cociente(N,2,C,R),dec\_bin(C,T),concatenar(T,[R],S),!.

/\*------------------------------------------------------------------\*/

/\*35-Calcula la potencia de un número, N elevado a la N1\*/

potencia(0,0,'No se puede calcular'):-!.

potencia(N,0,1):-N=\=0.

potencia(N,N1,S):-N2 is N1 -1,potencia(N,N2,R),!, S is R\*N.

/\*------------------------------------------------------------------\*/

/\*36-Transforma un número de binario a decimal\*/

bin\_dec([],0):-!.

bin\_dec([0|Y],S):-bin\_dec(Y,S).

bin\_dec([1|Y],S):-long(Y,N), potencia(2,N,T),bin\_dec(Y,R),!, S is T+R.

/\*------------------------------------------------------------------\*/

/\*37-Cuenta la cantidad de veces que ocurre un elemento en una lista en\*/

/\*todos los niveles\*/

ocurre(X,[],0):-!.

ocurre(X,[X|L],S):-ocurre(X,L,T),!,S is T+1.

ocurre(X,[Z|L],S):-lista(Z),ocurre(X,Z,T),ocurre(X,L,R),!,S is R+T.

ocurre(X,[Z|L],S):-X=\=Z,ocurre(X,L,S),!.

/\*------------------------------------------------------------------\*/

/\*38-Elimina elementos que se encuentran en la misma posici¢n en dos \*/

/\*lista, inclusive sublistas y devuelve las dos sublistas\*/

elim\_igual\_posic([],L,[],L):-!.

elim\_igual\_posic(L,[],L,[]):-!.

elim\_igual\_posic([X|M],[X|L],S1,S2):-elim\_igual\_posic(M,L,S1,S2).

elim\_igual\_posic([X|M],[Z|L],[X|S1],[Z|S2]):-not lista(Z), not lista(X)

,X=\=Z,elim\_igual\_posic(M,L,S1,S2).

elim\_igual\_posic([X|M],[Z|L],[X|S1],[Z|S2]):-elim\_igual\_posic(M,L,S1,S2),!.

/\*------------------------------------------------------------------\*/

/\*39-Elimina el elemento que se encuentra en la en‚sima posici¢n\*/

sacapos([X|Y],1,Y):-!.

sacapos([X|Y],N,[X|R]):-N1 is N-1, sacapos(Y,N1,R).

/\*------------------------------------------------------------------\*/

/\*40-Cuenta los elementos que se encuentran en dos listas en diferente \*/

/\*posiciones\*/

desord([],L,N,0).

desord([X|Y],L,P,S):-sacapos(L,P,L1), ocurre(X,L1,S1), P1 is P+1,

desord(Y,L,P1,R), S is S1 + R.

examina(L1,L2,D,S):-desord(L1,L2,1,D),orden(L1,L2,S).

/\*------------------------------------------------------------------\*/

/\*41-Reemplaza la aparici¢n de un elemento x en una lista, en todos los\*/

/\*niveles, por otro elemento y\*/

reemplazar(X,Y,[],[]):-!.

reemplazar(X,Y,[X|M],[Y|Z]):-reemplazar(X,Y,M,Z),!.

reemplazar(X,Y,[L|M],Z):-reemplazar(X,Y,L,T),reemplazar(X,Y,M,R),!,

concatenar([T],R,Z).

reemplazar(X,Y,[L|M],[L|Z]):-reemplazar(X,Y,M,Z),!.

/\*------------------------------------------------------------------\*/

/\*42-Da como resultado una lista, que es la intersecci¢n de las dos\*/

interseccion(L,[],[]):-!.

interseccion([],L,[]):-!.

interseccion([X|L],[X|H],[X|Z]):-interseccion(L,H,Z),!.

interseccion([X|L],[R|H],[X|Z]):-pert(X,H),elim\_prim\_pos(X,[R|H],L2),

interseccion(L,L2,Z),!.

interseccion([X|L],[R|H],Z):-interseccion(L,[R|H],Z),!.

/\*------------------------------------------------------------------\*/

/\*43-Elimina el primer elemento X que aparece en la lista\*/

elim\_prim\_pos(X,[],[]):-!.

elim\_prim\_pos(X,[X|M],M):-!.

elim\_prim\_pos(X,[R|M],[R|L]):-elim\_prim\_pos(X,M,L).

/\*------------------------------------------------------------------\*/

/\*44-Determina si un elemento X pertenece a una lista\*/

pert(X,[X|M]):-!.

pert(X,[R|M]):-pert(X,M).

/\*------------------------------------------------------------------\*/

/\*45-Genera una lista de tomos a partir de una lista anidada\*/

lista\_atomos([],[]):-!.

lista\_atomos([X|M],[X|Z]):-not lista(X), lista\_atomos(M,Z),!.

lista\_atomos([X|M],Z):-lista(X),lista\_atomos(X,L),lista\_atomos(M,M1),!,

concatenar(L,M1,Z).

/\*------------------------------------------------------------------\*/

/\*46-Arma dos listas con los elementos que se encuentran a la izquierda\*/

/\* y a la derecha del elemento x \*/

izq\_der\_x(X,[],[],[]):-!.

izq\_der\_x(X,[X|M],[],M):-!.

izq\_der\_x(X,[R|M],[R|L],F):-R=\=X,izq\_der\_x(X,M,L,F).

/\*------------------------------------------------------------------\*/

/\*47-Ordena una lista en orden ascendente\*/

ord\_ascend([],[]):-!.

ord\_ascend(L,R):-maximo(L,X),elim\_prim\_pos(X,L,L1),ord\_ascend(L1,L2),!,

concatenar(L2,[X],R).

/\*------------------------------------------------------------------\*/

/\*48-Ordena una lista en orden descendente\*/

ord\_descend(L,L2):-ord\_ascend(L,L3),invertir(L3,L2).

/\*------------------------------------------------------------------\*/

/\*49-Calcula el maximo elemento de una lista\*/

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

maximo([X,Y|M],X):-maximo([Y|M],Z),X>=Z.

maximo([X|M],Z):-maximo(M,Z),Z>X.

/\*------------------------------------------------------------------\*/

/\*50-Elimina las apariciones de X en el primer nivel\*/

elimina\_x([],X,[]):-!.

elimina\_x([X|M],X,Z):-elimina\_x(M,X,Z),!.

elimina\_x([R|M],X,[R|Z]):-elimina\_x(M,X,Z),!.

/\*------------------------------------------------------------------\*/

/\*51-Elimina una sublista L de la lista L1 \*/

elim\_subl([],L,[]):-!.

elim\_subl(L1,L,Z):-long(L,N),da\_n\_pri(L1,N,L2),L2=L,sac\_n\_pri(L1,N,L3),

elim\_subl(L3,L,Z),!.

elim\_subl([X|Y],L,[X|Z]):-elim\_subl(Y,L,Z),!.

/\*------------------------------------------------------------------\*/

/\*52-Determina si dos conjuntos son iguales\*/

conj\_igual([],[]):-!.

conj\_igual([X|Y],L):-pert(X,L),elim\_prim\_pos(X,L,L1), conj\_igual(Y,L1).

conj\_igual([X|Y],L):-lista(X), sublista(X,L),elim\_subl(X,L,L1),conj\_igual(Y,L1).

/\*------------------------------------------------------------------\*/

/\*53-Arma una lista de n elementos, los mayores de la lista\*/

n\_mayores(L1,N,L):-ord\_descend(L1,L2),da\_n\_pri(L2,N,L).

/\*------------------------------------------------------------------\*/

/\*54-Devuelve una lista con la diferencia entre dos conjuntos\*/

dif\_conj(L,[],L):-!.

dif\_conj([],L,L):-!.

dif\_conj([X|Y],L,[X|P]):-not pert(X,L), dif\_conj(Y,L,P),!.

dif\_conj([X|Y],L,P):-pert(X,L),elim\_prim\_pos(X,L,L1),dif\_conj(Y,L1,P),!.

/\*------------------------------------------------------------------\*/

/\*55-Inserta un elemento en la posicion en‚sima\*/

insertar(L,1,X,[X|L]):-!.

insertar([X|Y],N,R,[X|L]):-N1 is N-1, insertar(Y,N1,R,L).

/\*------------------------------------------------------------------\*/

/\*56-Calcula la posicion del último elemento n\*/

posult(L,X,N):-invertir(L,L1), prim\_pos(X,L1,T), long(L1,M),N is M-T.

/\*------------------------------------------------------------------\*/

/\*57-Reemplaza el elemento en‚simo\*/

reemp\_n([X|Y],1,L,[L|Y]):-!.

reemp\_n([X|Y],N,L,[X|Y1]):-N1 is N-1, reemp\_n(Y,N1,L,Y1).

/\*------------------------------------------------------------------\*/

/\*58-Arma una lista de N elementos a partir de la posici¢n P\*/

tomarN\_desd\_P(L,0,N,L1):-da\_n\_pri(L,N,L1).

tomarN\_desd\_P([X|Y],P,N,L1):-P1 is P-1,tomarN\_desd\_P(Y,P1,N,L1).

/\*------------------------------------------------------------------\*/

/\*59-Determina si un par de elementos son consecutivos\*/

consecut([X,Y|M],X,Y):-!.

consecut([R|M],X,Y):-consecut(M,X,Y).

/\*------------------------------------------------------------------\*/

/\*60-Arma una lista con los elementos duplicados que se encuentran en \*/

/\*una lista\*/

lista\_dup([],[]):-!.

lista\_dup([X|L],[X|Y]):-pert(X,L),elim\_prim\_pos(X,L,L1),lista\_dup(L1,Y),!.

lista\_dup([X|L],Y):-lista\_dup(L,Y),!.

/\*------------------------------------------------------------------\*/

/\*61-Arma una lista con todos los elementos que se encuentran dentro del\*/

/\*intervalo\*/

interval([],N1,N2,[]):-!.

interval([X|Y],N1,N2,[X|L]):-X>=N1,X=<N2,interval(Y,N1,N2,L),!.

interval([X|Y],N1,N2,L):-interval(Y,N1,N2,L),!.

/\*------------------------------------------------------------------\*/

/\*62-Determina si una lista se encuentra ordenada segun, m ximo, m¡nimo,\*/

/\*meseta o sin orden\*/

identif(X,maximo):- esmaximo(X),!.

identif(X,minimo):- esminimo(X),!.

identif(X,meseta):- esmeseta(X),!.

identif(X,sinorden):-!.

esmaximo([X,Y|Z]):- X<Y,

esmaximo([Y|Z]).

esmaximo([X,Y|Z]):- X>Y,

decreciente([Y|Z]).

esminimo([X,Y|Z]):- X>Y,

esminimo([Y|Z]).

esminimo([X,Y|Z]):- X<Y,

creciente([Y|Z]).

esmeseta([X,Y,Z|W]):- X<Y,

esmeseta([Y,Z|W]).

esmeseta([X,Y,Z|W]):- X=Y,Y=Z,

esmeseta([Y,Z|W]).

esmeseta([X,Y,Z|W]):- X=Y,Y>Z,

decreciente([Z|W]).

/\*------------------------------------------------------------------\*/

/\*63-Calcula una dieta, ejercicio que est en un examen\*/

alimento(a,posillo\_zucaritas,110).

alimento(b,vegetal,155).

alimento(c,posillo\_de\_leche,660).

alimento(c,huevo\_frito,110).

alimento(c,bife\_de\_vaca,185).

alimento(c,pechuga\_de\_pollo,185).

alimento(d,helado,330).

plato([1,B,C]):-alimento(A,B,C).

calorias([A,B,C],C):-!.

pertenece(B,[X|L]):- pertenece1(B,X).

pertenece(B,[X|L]):-pertenece(B,L).

pertenece1([V,B,M],[A,B,C]):-!.

suma([],0):-!.

suma([X|L],S):- suma(L,T), calorias(X,R), S is R+T.

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

insertar(X,L,[X|L]):-not pertenece(X,L),!.

insertar(X,L,R):-suma\_c(X,L,R).

suma\_c(X,[R|L],T):-pertenece1(X,R),suma\_caloria(X,R,U),concatenar([U],L,T).

suma\_c(X,[R|L],[R|T]):-suma\_c(X,L,T).

suma\_caloria([A,B,C],[R,B,V],[X,B,N]):-N is V+C, X is R+A.

concatenar([],L,L):-!.

concatenar([X|R],L,[X|T]):-concatenar(R,L,T).

arma\_dieta(L,L):-suma(L,T),T>2400,T<2800,!.

arma\_dieta(L,L1):-suma(L,T),T<2400,plato(X),insertar(X,L,L2),arma\_dieta(L2,L1).

arma\_dieta1(L,N):-arma\_dieta([],L),suma(L,N).

/\*------------------------------------------------------------------\*/

/\*64-Arma una lista con las funciones que son invocadas por una funci¢n\*/

invoca(a,[b,c,d]):-!.

invoca(b,[e]):-!.

invoca(d,[f,g,h]):-!.

invoca(X,[]):-!.

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

invocar([X|Y],S):-invoca(X,T), invocar(Y,N), concatenar([X],[T],D),

concatenar([D],N,S).

invoca-a([],[]):-!.

invoca-a(X,S):-invoca(X,P), invocar(P,T), concatenar([X],T,S).

/\*------------------------------------------------------------------\*/

/\*65-Planifica las actividades para la semana, ejercicio de examen\*/

genera\_univ([universidad,[19,24],lunes]).

genera\_univ([universidad,[19,24],miercoles]).

genera\_dep([tennis,[10,12],lunes]).

genera\_dep([tennis,[10,12],jueves]).

genera\_dep([jogging,[12.3,13],lunes]).

genera\_idio([ingles,[10.3,12],lunes]).

genera\_idio([ingles,[10.3,12],miercoles]).

genera\_idio([frances,[8,10],lunes]).

genera\_idio([frances,[8,10],miercoles]).

act\_posb([A,B,C]):-genera\_dep(A),genera\_univ(B),genera\_idio(C).

posible([],N):-!.

posible([[A,[B,C],D]|M],[R,[F,G],D]):-F>=C,posible(M,[R,[F,G],D]).

posible([[A,[B,C],D]|M],[R,[F,G],D]):-G=<B,posible(M,[R,[F,G],D]).

posible([[A,[B,C],D]|M],[R,[F,G],L]):-L\==D,posible(M,[R,[F,G],L]).

compara([A,B,C]):-posible([B,C],A), posible([A,C],B),posible([B,A],C).

plan(A):-act\_posb(A),compara(A).

/\*------------------------------------------------------------------\*/

/\*66-Transforma una lista en formato infijo a formato prefijo\*/

transforma([Op,Op1,Op2],[A,Op,B]):-lista(Op1),lista(Op2), transforma(Op1,A),

transforma(Op2,B).

transforma([A,B,C],[B,A,C]):-not lista(B),not lista(C),!.

/\*------------------------------------------------------------------\*/

/\*67-Ejercicio de examen, decodifica un c¢digo\*/

diccionario([codigo([a,b,f],[0,1]),codigo([t,h],[1]),codigo([w,w,a],[1,1])]).

buscar\_cod(L,[codigo(L,L2)|M],L2):-!.

buscar\_cod(L,[X|M],L2):-buscar\_cod(L,M,L2),!.

decodifica([],N,[]):-!.

decodifica(L,N,S):-da\_n\_pri(L,N,L1),diccionario(L5),buscar\_cod(L1,L5,L2),

sac\_n\_pri(L,N,L3),decodifica(L3,1,L4),!,concatenar(L2,L4,S).

decodifica(L,N,S):-N1 is N+1, decodifica(L,N1,S).

/\*------------------------------------------------------------------\*/

/\*68-Ordenamiento quicksort de una lista\*/

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

quicksort([X|Y],L):-obtenerlista(X,Y,Lme,Lma),quicksort(Lme,T),

quicksort(Lma,R), concatenar(T,[X|R],L).

/\*------------------------------------------------------------------\*/

/\*69-Sumador de numeros binarios\*/

and(0,0,0).

and(1,0,0).

and(0,1,0).

and(1,1,1).

or(0,0,0).

or(1,0,1).

or(0,1,1).

or(1,1,1).

xor(0,0,0).

xor(1,0,1).

xor(0,1,1).

xor(1,1,0).

semisum(A,B,Bs,Ba):-and(A,B,Ba),xor(A,B,Bs).

sumcomp(A,B,Ae,Bs,Ba):-semisum(A,B,Bs1,Ba1),semisum(Ae,Bs1,Bs,Ba2),

or(Ba1,Ba2,Ba).

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

ultimo(L,U):-invertir(L,[U|R]).

elim\_ulti([X],[]):-!.

elim\_ulti(L,L1):-invertir(L,[U|L2]),invertir(L2,L1).

sumador(B1,[],0,B1):-!.

sumador(B1,[],1,B2):-sumador(B1,[1],0,B2).

sumador([],B1,0,B1):-!.

sumador([],B1,1,B2):-sumador([1],B1,0,B2).

sumador(B1,B2,Ba,Bs):-ultimo(B1,Bs1),ultimo(B2,Bs2), sumcomp(Bs1,Bs2,Ba,X,Y),

elim\_ulti(B1,B3),elim\_ulti(B2,B5), sumador(B3,B5,Y,BsAux),

concatenar(BsAux,[X],Bs).

sumadorbinario(B1,B2,B3):-sumador(B1,B2,0,B3),!.

/\*------------------------------------------------------------------\*/

/\*70-Sumador de números binarios (haci‚ndose el b...)\*/

sumadorbin(B1,B2,B3):-bin\_dec(B1,S),bin\_dec(B2,S1),R is S+S1,dec\_bin(R,B3).

/\*------------------------------------------------------------------\*/

/\*71-Determina si un árbol es vacio\*/

vacio(aárbol(nil)):-!.

/\*------------------------------------------------------------------\*/

/\*72-Arma una lista con todos los elementos del árbol\*/

elem\_aárbol(tree(A,nil,nil),[A]):-!.

elem\_aárbol(tree(A,X,nil),S):-elem\_aárbol(X,P), concatenar([A],P,S).

elem\_aárbol(tree(A,nil,X),S):-elem\_aárbol(X,P), concatenar([A],P,S).

elem\_aárbol(tree(A,X,Y),S):-elem\_aárbol(X,P),elem\_aárbol(Y,R),concatenar([A],P,U),

concatenar(U,R,S).

/\*------------------------------------------------------------------\*/

/\*73-Cuenta la cantidad de elementos que posee el árbol\*/

cant\_elem(tree(A,nil,nil),1):-!.

cant\_elem(tree(A,X,nil),N):-cant\_elem(X,K),N is 1 + K.

cant\_elem(tree(A,nil,X),N):-cant\_elem(X,K),N is 1 + K.

cant\_elem(tree(A,X,Y),N):-cant\_elem(X,K),cant\_elem(Y,T), N is 1 + K + T.

/\*------------------------------------------------------------------\*/

/\*74-Determina si un árbol es completo\*/

aárbol\_comp(tree(A,nil,nil)):-!.

aárbol\_comp(tree(A,X,Y)):-porfundidad(X,N),profundidad(Y,M),

N=:=M,aárbol\_comp(X),aárbol\_comp(Y),!.

/\*------------------------------------------------------------------\*/

/\*75-Calcula la profundidad del árbol\*/

profundidad(tree(A,nil,nil),0):-!.

profundidad(tree(A,Y,nil),N):-profundidad(Y,B),N is 1 + B.

profundidad(tree(A,nil,X),N):-profundidad(X,B),N is 1 + B.

profundidad(tree(A,Y,X),N):-profundidad(X,B),profundidad(Y,C), mayor(B,C,N).

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

mayor(B,C,B).

/\*------------------------------------------------------------------\*/

/\*76-Recorre un árbol en preorden\*/

preorden(aárbol(A,nil,nil),[A]):-!.

preorden(aárbol(A,X,nil),[A|S]):-preorden(X,S),!.

preorden(aárbol(A,nil,X),[A|S]):-preorden(X,S),!.

preorden(aárbol(A,X,Y),[A|S]):-preorden(X,T),preorden(Y,O),!,

concatenar(T,O,S).

/\*------------------------------------------------------------------\*/

/\*77-Recorre un árbol en inorden\*/

inorden(aárbol(A,nil,nil),[A]):-!.

inorden(aárbol(A,X,nil),S):-inorden(X,C),concatenar(C,[A],S).

inorden(aárbol(A,nil,X),[A|S]):-inorden(X,S).

inorden(aárbol(A,X,Y),S):-inorden(X,C),inorden(Y,F),concatenar(C,[A],D),

concatenar(D,F,S).

/\*------------------------------------------------------------------\*/

/\*78-Recorre un árbol en posorden\*/

posorden(aárbol(A,nil,nil),[A]):-!.

posorden(aárbol(A,X,nil),S):-posorden(X,C),concatenar(C,[A],S).

posorden(aárbol(A,nil,X),S):-posorden(X,C),concatenar(C,[A],S).

posorden(aárbol(A,X,Y),S):-posorden(X,C),posorden(Y,F),concatenar(C,F,D),

concatenar(D,[A],S).

/\*------------------------------------------------------------------\*/

/\*79-Determina si un elemento pertenece a un árbol\*/

pert\_aárbol(aárbol(A,X,Y),A):-!.

pert\_aárbol(aárbol(A,X,Y),B):-pert\_aárbol(X,B),!.

pert\_aárbol(aárbol(A,X,Y),B):-pert\_aárbol(Y,B),!.

/\*------------------------------------------------------------------\*/