word(astante, a,s,t,a,n,t,e).

word(astoria, a,s,t,o,r,i,a).

word(baratto, b,a,r,a,t,t,o).

word(cobalto, c,o,b,a,l,t,o).

word(pistola, p,i,s,t,o,l,a).

word(statale, s,t,a,t,a,l,e).

crossword(V1,V2,V3,H1,H2,H3):- word(V1, \_,L1,\_,L2,\_,L3,\_),

word(V2, \_,L4,\_,L5,\_,L6,\_), word(V3, \_,L7,\_,L8,\_,L9,\_),

word(H1, \_,L1,\_,L4,\_,L7,\_), word(H2, \_,L2,\_,L5,\_,L8,\_),

word(H3, \_,L3,\_,L6,\_,L9,\_), V1\==V2, V1\==V3, V1\==H1,

V1\==H2, V1\==H3, V2\==V3, V2\==H1, V2\==H2, V2\==H3,

V3\==H1, V3\==H2, V3\==H3, H1\==H2, H1\==H3, H2\==H3.

add(0,X,X).

add(succ(X),Y,succ(Z)):-add(X,Y,Z).

child(anne,bridget).

child(bridget,caroline).

child(caroline,donna).

child(donna,emily).

descend(X,Y) :- child(X,Y).

/\*descend(X,Y) :- descend(Z,Y), child(X,Z).\*/

descend(X,Y) :- descend(X,Z), descend(Z,Y).

directTrain(saarbruecken,dudweiler).

directTrain(forbach,saarbruecken).

directTrain(freyming,forbach).

directTrain(stAvold,freyming).

directTrain(fahlquemont,stAvold).

directTrain(metz,fahlquemont).

directTrain(nancy,metz).

travel(X,X).

travel(X,Y):- directTrain(X,Y).

travel(X,Y):- directTrain(X,Z), travel(Z,Y).

greater\_than(\_,0).

greater\_than(succ(X),succ(Y)):- greater\_than(X,Y).

linie(0,\_):-nl.

linie(X,Y):- X>0, write(Y),X2 is X-1,linie(X2,Y).

dreptunghi(\_,0,\_):-nl.

dreptunghi(N,X,Y):-linie(N,Y),X>0,X2 is X-1,dreptunghi(N,X2,Y).

patrat(0,\_).

patrat(X,Y):-dreptunghi(X,X,Y).

byCar(auckland,hamilton).

byCar(hamilton,raglan).

byCar(valmont,saarbruecken).

byCar(valmont,metz).

byTrain(metz,frankfurt).

byTrain(saarbruecken,frankfurt).

byTrain(metz,paris).

byTrain(saarbruecken,paris).

byPlane(frankfurt,bangkok).

byPlane(frankfurt,singapore).

byPlane(paris,losAngeles).

byPlane(bangkok,auckland).

byPlane(singapore,auckland).

byPlane(losAngeles,auckland).

travel2(X,X).

travel2(X,Y):- byCar(X,Y).

travel2(X,Y):- byPlane(X,Y).

travel2(X,Y):- byTrain(X,Y).

travel2(X,Y):- byCar(X,Z),travel2(Z,Y).

travel2(X,Y):- byTrain(X,Z),travel2(Z,Y).

travel2(X,Y):- byPlane(X,Z),travel2(Z,Y).

travel3(X,Y,Z):-travel2(X,Y).

/\*travel2(X,Y):- travel2(X,Z),travel2(Z,Y).

travel(X,Y):- byCar(X,Z),byPlane(Z,Y).

travel(X,Y):- byCar(X,Z),byTrain(Z,Y).

travel(X,Y):- byTrain(X,Z),byPlane(Z,Y).

travel(X,Y):- byTrain(X,Z),byCar(Z,Y).

travel(X,Y):- byPlane(X,Z),byTrain(Z,Y).

travel(X,Y):- byPlane(X,Z),byCar(Z,Y).

travel(X,Y):- byCar(X,W), byPlane(W,Z), byTrain(Z,Y).\*/

LISTE

member(X,[X|\_]).

member(X,[H|T]):-member(X,T),H\==X.

indice(X,[X|\_],1).

indice(X,[H|T],I2):- indice(X,T,I), I2 is I+1, H\==X.

a2b([],[]).

a2b([a|T1],[b|T2]):-a2b(T1,T2).

/\*second(X,[\_|X]).\*/

second(X,[\_|[X|\_]]).

swap12([],[]).

swap12([H1|[H2|T]],[H2|[H1|T]]).

tran(eins,one).

tran(zwei,two).

tran(drei,three).

tran(vier,four).

tran(fuenf,five).

tran(sechs,six).

tran(sieben,seven).

tran(acht,eight).

tran(neun,nine).

listtran([],[]).

listtran([H|T],[X|Y]):- tran(H,X), listtran(T,Y).

twice([],[]).

twice([H|T],[H|[H|Y]]):- twice(T,Y).

combine1([],[],[]).

combine1([H1|T1],[H2|T2],[H1|[H2|Y]]):- combine1(T1,T2,Y).

combine2([],[],[]).

combine2([H1|T1],[H2|T2],[[H1|H2]|Y]):- combine2(T1,T2,Y).

combine3([],[],[]).

combine3([H1|T1],[H2|T2],[j(H1,H2)|Y]):- combine3(T1,T2,Y).

maxx([],X,X).

maxx([H|T],X,M):- H>X, maxx(T,H,M).

maxx([H|T],X,M):- H=<X,maxx(T,X,M).

max(L,M):-maxx(L,0,M).

greater(X,W):- X>=W.

suma(A,B,C):- C =:= A+B.

addone([],[]).

addone([H|T],[X|Y]):- X is H+1, addone(T,Y).

scmult(\_,[],[]).

scmult(X,[H|T],[Y|Z]):- Y is X\*H, scmult(X,T,Z).

dot([],[],0).

dot([H|T],[A|B],X):- dot(T,B,Y), X is H\*A + Y.

Liste2

concat([],L,L).

concat([H|T],L2,[H|L3]):- concat(T,L2,L3).

prefix(P,L):- concat(P,\_,L).

sufix(S,L):- concat(\_,S,L).

sublist(Subl,L):- sufix(S,L), prefix(Subl,S).

egal([],[]).

egal([H|T],[H|B]):- egal(T,B).

double(L):-prefix(P,L),append(P,P,L).

palindrome(L):- rev(L,R),egal(L,R).

scotprim([\_|T],T).

scotult(L,R):- rev(L,X),scotprim(X,Y),rev(Y,R).

toptail([\_|T],R):- scotult(T,R).

lastrec([H],H).

lastrec([\_|T],X):-lastrec(T,X).

first([H|\_],H).

last(L,X):- first(R,X),rev(L,R).

comp([H|T],[A|B]):- last([A|B],H),last([H|T],A).

swap2(L1,L2):- toptail(L1,X), toptail(L2,Y), egal(X,Y),

comp(L1,L2).