```
reuniune(A,B,R) :- setof(X, member(X,A); member(X,B), R), !.
reuniune(_,_,[]).

intersectie(A,B,I) :- setof(X, (member(X,A), member(X,B)), I), !.
intersectie(_,_,[]).

prodcart(A,B,P) :- setof((X,Y), (member(X,A), member(Y,B)), P), !.
prodcart(_,_,[]).

sterge(_,[],[]).
sterge(H,[H|T],L) :- !, sterge(H,T,L).
sterge(X,[H|T],[H|L]) :- sterge(X,T,L).

elimdupl([],[]).
elimdupl([H|T],[H|L]) :- sterge(H,T,M), elimdupl(M,L).

sublista([],_).
sublista([H|T],[H|L]) :- sublista(T,L).
sublista([H|T],[_|L]) :- sublista([H|T],L).

sublistele(L,LS) :- setof(S, sublista(S,L), LS).

inclusa([],_).
inclusa([H|T],M) :- member(H,M), inclusa(T,M).

egaldemult(L,M) :- inclusa(L,M), inclusa(M,L).
```

```
egalmult(L,M) :- permutare(L,M).

stergeuna(_,[],_) :- fail.
stergeuna(H,[H|T],T).
stergeuna(X,[H|T],[H|L]) :- stergeuna(X,T,L).

permutare([],[]).
permutare([H|T],P) :- permutare(T,Q), stergeuna(H,P,Q).

% F:P(T)->P(A)xP(B)

egalperechimult((A,B),(C,D)) :- egaldemult(A,C), egaldemult(B,D).

injectiva(F) :- not((member((X,(Y,Z)),F), member((U,(V,W)),F),
                egalperechimult((Y,Z),(V,W)), not(egaldemult(X,U)))).

surjectiva(F,Codom) :- not((member(Per,Codom),
         not((member((_,AltaPer),F), egalperechimult(Per,AltaPer))))).

implica(P,Q) :- not(P);Q.
echiv(P,Q) :- implica(P,Q), implica(Q,P).

functiaF(F,T,A,B) :- setof((X,(XintersA,XintersB)), (sublista(X,T),
        intersectie(X,A,XintersA), intersectie(X,B,XintersB)), F).

verif(T) :- not((sublista(A,T), sublista(B,T), functiaF(F,T,A,B),
        write(A), tab(1), write(B), nl,
```

```
        sublistele(A,PA), sublistele(B,PB), prodcart(PA,PB,P),
   not((echiv(injectiva(F), (reuniune(A,B,R), elimdupl(T,U), egalmult(R,U))),
        echiv(surjectiva(F,P), (intersectie(A,B,I), I=])))))).

lista(0,[]).
lista(N,[N|L]) :- N>0, PN is N-1, lista(PN,L).

verifmult(T) :- not((sublista(A,T), sublista(B,T), functiaF(F,T,A,B),
        write(A), tab(1), write(B), nl,
        sublistele(A,PA), sublistele(B,PB), prodcart(PA,PB,P),
   not((echiv(injectiva(F), (reuniune(A,B,R), egalmult(R,T))),
        echiv(surjectiva(F,P), (intersectie(A,B,I), I=])))))).

verifPanaLaCard(N) :- (N>0, !, PN is N-1, verifPanaLaCard(PN) ; N>=0),
        lista(N,T), write('T='), write(T), nl, verifmult(T),
                    write('----------------------------------------'), nl.

verifpanalaCard(-1).
verifpanalaCard(N) :- N>=0, !, PN is N-1, verifpanalaCard(PN),
        lista(N,T), write('T='), write(T), nl, verifmult(T),
                    write('----------------------------------------'), nl.

cale('d:/tempwork/').
numeFis('verificare.txt').

verifpanalacard(N) :- cale(C), numeFis(Fis), atom_concat(C,Fis,Fisierul),
                tell(Fisierul), verifpanalaCard(N), told.
```

```
verifpanalacardinalul(-1).
verifpanalacardinalul(N) :- N>=0, !, PN is N-1, verifpanalacardinalul(PN),
lista(N,T), cale(C), atom_concat('verifptcardinalul',N,Fis),
        atom_concat(Fis,'.txt',Fisier), atom_concat(C,Fisier,Fisierul),
        tell(Fisierul), write('T='), write(T), nl, verifmult(T), told.

listax(0,[]).
listax(N,[Atom|L]) :- N>0, atom_concat(x,N,Atom), PN is N-1, listax(PN,L).

verifpanalacardinalulx(-1).
verifpanalacardinalulx(N) :- N>=0, !, PN is N-1, verifpanalacardinalulx(PN),
listax(N,T), cale(C), atom_concat('verifptcardinalcux',N,Fis),
        atom_concat(Fis,'.txt',Fisier), atom_concat(C,Fisier,Fisierul),
        tell(Fisierul), write('T='), write(T), nl, verifmult(T), told.

numeFisier('ptcardcux').

verifpanalacardinal(-1,_).
verifpanalacardinal(N,Extensie) :- N>=0, !, PN is N-1,
        verifpanalacardinal(PN,Extensie), listax(N,T),
        cale(Cale), numeFisier(Fi), atom_concat(Fi,N,Fis),
        atom_concat(Fis,Extensie,Fisier), atom_concat(Cale,Fisier,Fisierul),
        tell(Fisierul), write('T='), write(T), nl, verifmult(T), told.

/* Interogati:
?- verifpanalacardinal(5,'.txt').
```

```
?- verifpanalacardinal(50,'.docx').
*/

verifMult(T) :- sublistele(T,PT), prodcart(PT,PT,Prod), auxverif(T,Prod,0).

auxverif(_,[],_).
auxverif(T,[(A,B)|Coada],N) :- functiaF(F,T,A,B),
        write(A), tab(1), write(B), tab(1), SN is N+1, write(SN), nl,
        sublistele(A,PA), sublistele(B,PB), prodcart(PA,PB,P),
        echiv(injectiva(F), (reuniune(A,B,R), egalmult(R,T))),
        echiv(surjectiva(F,P), (intersectie(A,B,I), I=[])),
        auxverif(T,Coada,SN).

verifpanalacardcunr(-1).
verifpanalacardcunr(N) :- N>=0, !, PN is N-1, verifpanalacardcunr(PN),
listax(N,T), cale(C), atom_concat('verifptcardcux',N,Fis),
        atom_concat(Fis,'.txt',Fisier), atom_concat(C,Fisier,Fisierul),
        tell(Fisierul), write('T='), write(T), nl, verifMult(T), told.

/* Verificari intermediare, pentru corectarea predicatelor:

?- functiaF(F,[],[],[]), (injectiva(F), !, write(F), write(' e injectiva'); write(F),
write(' nu e injectiva')), nl, ((reuniune([],[],R), elimdupl([],U), egalmult(R,U)), !,
write('AUB=T'); write('AUB=/=T')).

?- functiaF(F,[],[],[]), sublistele([],PA), write(PA), tab(1), sublistele([],PB), write(PB),
nl, prodcart(PA,PB,P), write(P), nl, (surjectiva(F,P), !, write(F), write(' e surjectiva');
```

```
write(F), write(' nu e surjectiva')), nl, (intersectie([],[],I), I=[], !, write('A^B=0');
write('A^B=/=0')).

?- functiaF(F,[a],[],[]), (injectiva(F), !, write(F), write(' e injectiva'); write(F),
write(' nu e injectiva')), nl, ((reuniune([],[],R), elimdupl([a],U), egalmult(R,U)), !,
write('AUB=T'); write('AUB=/=T')).

?- functiaF(F,[a],[],[]), sublistele([],PA), write(PA), tab(1), sublistele([],PB),
write(PB), nl, prodcart(PA,PB,P), write(P), nl, (surjectiva(F,P), !, write(F), write(' e
surjectiva'); write(F), write(' nu e surjectiva')), nl, (intersectie([],[],I), I=[], !,
write('A^B=0'); write('A^B=/=0')).

?- functiaF(F,[a],[],[a]), (injectiva(F), !, write(F), write(' e injectiva'); write(F),
write(' nu e injectiva')), nl, ((reuniune([],[a],R), elimdupl([a],U), egalmult(R,U)), !,
write('AUB=T'); write('AUB=/=T')).

?- functiaF(F,[a],[],[a]), sublistele([],PA), write(PA), tab(1), sublistele([a],PB),
write(PB), nl, prodcart(PA,PB,P), write(P), nl, (surjectiva(F,P), !, write(F), write(' e
surjectiva'); write(F), write(' nu e surjectiva')), nl, (intersectie([],[a],I), I=[], !,
write('A^B=0'); write('A^B=/=0')).

?- functiaF(F,[a],[a],[]), (injectiva(F), !, write(F), write(' e injectiva'); write(F),
write(' nu e injectiva')), nl, ((reuniune([a],[],R), elimdupl([a],U), egalmult(R,U)), !,
write('AUB=T'); write('AUB=/=T')).

?- functiaF(F,[a],[a],[]), sublistele([a],PA), write(PA), tab(1), sublistele([],PB),
write(PB), nl, prodcart(PA,PB,P), write(P), nl, (surjectiva(F,P), !, write(F), write(' e
```

```
surjectiva'); write(F), write(' nu e surjectiva')), nl, (intersectie([a],[],I), I=[], !,
write('A^B=0'); write('A^B=/=0')).

?- functiaF(F,[a],[a],[a]), (injectiva(F), !, write(F), write(' e injectiva'); write(F),
write(' nu e injectiva')), nl, ((reuniune([a],[a],R), elimdupl([a],U), egalmult(R,U)), !,
write('AUB=T'); write('AUB=/=T')).

?- functiaF(F,[a],[a],[a]), sublistele([a],PA), write(PA), tab(1), sublistele([a],PB),
write(PB), nl, prodcart(PA,PB,P), write(P), nl, (surjectiva(F,P), !, write(F), write(' e
surjectiva'); write(F), write(' nu e surjectiva')), nl, (intersectie([a],[a],I), I=[], !,
write('A^B=0'); write('A^B=/=0')).

?- functiaF(F,[2,1],[],[2,1]), (injectiva(F), !, write(F), write(' e injectiva'); write(F),
write(' nu e injectiva')), nl, ((reuniune([],[2,1],R), elimdupl([2,1],U), egalmult(R,U)), !,
write('AUB=T'); write('AUB=/=T')).

?- functiaF(F,[2,1],[],[2,1]), sublistele([],PA), write(PA), tab(1), sublistele([2,1],PB),
write(PB), nl, prodcart(PA,PB,P), write(P), nl, (surjectiva(F,P), !, write(F), write(' e
surjectiva'); write(F), write(' nu e surjectiva')), nl, (intersectie([],[2,1],I), I=[], !,
write('A^B=0'); write('A^B=/=0')).
*/
```