```
:- [lab4lmc1].

% Compunerea de relatii binare:

comprel(S,R,SoR) :- setof((X,Z), Y^(member((X,Y),R),member((Y,Z),S)), SoR), !.
comprel(_,_,[]).

% Produsul a doua relatii binare:

prodrel(R,Q,RxQ) :- setof(((X,Y),(U,V)),
                (member((X,Y),R), member((U,V),Q)), RxQ), !.
prodrel(_,_,[]).

% Exemple de relatii binare, pentru a fi folosite cu predicatele de mai sus:

relP([(a,b),(a,c),(a,d),(a,e),(b,c),(b,d)]).
relQ([(a,b),(a,c),(b,d),(c,e),(e,f)]).
relR([(a,b),(a,c)]).
relS([(b,d)]).
relT([(a,a),(a,c),(b,a),(b,b),(b,d),(c,b),(c,c),(d,a),(d,d)]).
relU([(a,a),(a,b),(b,a),(b,b),(c,c),(c,d),(d,c),(d,d),(e,e)]).
relV([(a,a),(a,b),(a,c),(b,b),(b,c),(c,c),(d,d),(d,e),(e,e)]).

/* Interogati:
?- relR(R), relS(S), comprel(S,R,SoR).
?- relR(R), relS(S), comprel(R,S,RoS).
?- relR(R), relS(S), prodrel(R,S,RxS).
```

```
In afisarea lui RxS, Prolog-ul elimina parantezele din jurul fiecarei a doua perechi din
perechile de perechi din RxS. */

% Relatie binara R pe o multime A:

relbinara(R,A) :- relbin(R,A,A).

% Diagonala multimii A, cu setof, apoi recursiv:

diag(A,D) :- setof((X,X), member(X,A), D).

diagonala([],[]).
diagonala([H|T],[(H,H)|L]) :- diagonala(T,L).
```

```
% Puterile intregi nenule ale unei relatii binare pe o multime:

putere(R,1,R).
putere(R,N,RlaN) :- N>1, PN is N-1, putere(R,PN,RlaPN), comprel(RlaPN,R,RlaN).
putere(R,N,RlaN) :- N<0, ModulN is -N, invrel(R,I), putere(I,ModulN,RlaN).

/* Puterile intregi ale unei relatii binare pe o multime A: urmatorul predicat, de aritate
4, nu va fi confundat de Prolog cu predicatul ternar de mai sus: */

putere(_,A,0,D) :- diag(A,D).
putere(R,A,N,RlaN) :- N>0, PN is N-1, putere(R,A,PN,RlaPN),
                          comprel(RlaPN,R,RlaN).
putere(R,A,N,RlaN) :- N<0, ModulN is -N, invrel(R,I), putere(I,A,ModulN,RlaN).
```

```
/* Interogati:
?- diag([a,b,c],D).
?- diagonala([a,b,c],D).
?- relQ(Q), putere(Q,1,Qla1), putere(Q,2,Qla2), putere(Q,3,Qla3), putere(Q,4,Qla4).
?- relQ(Q), putere(Q,2,Qla2), putere(Q,-2,QlaMinus2).
?- relQ(Q), putere(Q,[a,b,c,d,e,f,g],0,Qla0), putere(Q,[a,b,c,d,e,f,g],1,Qla1).
?- relQ(Q), putere(Q,[a,b,c,d,e,f,g],2,Qla2), putere(Q,[a,b,c,d,e,f,g],-2,QlaMinus2).
?- relQ(Q), putere(Q,[a,b,c,d,e,f,g],-3,QlaMinus3).
*/

/* Testarea reflexivitatii unei relatii binare pe o multime A, cu negatie, apoi recursiv: */

refl(R,A) :- not((member(X,A), not(member((X,X),R)))).

reflexiva(_,[]).
reflexiva(R,[H|T]) :- member((H,H),R), reflexiva(R,T).

/* Testarea simetriei unei relatii binare pe o multime, cu negatie, apoi recursiv: */

sim(R) :- not((member((X,Y),R), not(member((Y,X),R)))).

simetrica(R) :- auxsim(R,R).

auxsim(_,[]).
auxsim(R,[(X,Y)|T]) :- member((Y,X),R), auxsim(R,T).
```

```
% Testarea tranzitivitatii unei relatii binare pe o multime:

tranz(R) :- not((member((X,Y),R), member((Y,Z),R), not(member((X,Z),R)))).

/* Interogati:
?- relP(P), refl(P,[a,b,c,d]).
?- relP(P), reflexiva(P,[a,b,c,d]).
?- relT(T), refl(T,[a,b,c,d]).
?- relT(T), reflexiva(T,[a,b,c,d]).
T={(a,a),(a,c),(b,a),(b,b),(b,d),(c,b),(c,c),(d,a),(d,d)}, relatie binara reflexiva pe
multimea {a,b,c,d}, este solutie pentru cele doua interogari precedente. Desigur, putem
interoga:
?- refl([(a,a),(a,c),(b,a),(b,b),(b,d),(c,b),(c,c),(d,a),(d,d)],[a,b,c,d]).
La fel mai jos. Interogati:
?- relR(R), sim(R).
?- relR(R), simetrica(R).
?- relU(U), sim(U).
?- relQ(Q), tranz(Q).
?- relS(S), tranz(S).
Observati ca nu exista drumuri de lungime 2 formate cu arce din S={(b,d)}, asadar S este in
mod trivial tranzitiva. Aici b,d sunt diferite, fiind constante Prolog cu nume diferite.
Daca ar fi egale, S ar fi in continuare tranzitiva. Interogati:
?- tranz([(b,b)]).
Sa vedem ce fel de relatii binare pe o multime A sunt cele memorate cu predicatele unare de
mai sus: */

tiprel(R,A) :- write(R), write(':'), nl, tab(3),
```

```
        (refl(R,A), !, write('e reflexiva pe multimea '), write(A);
        write('nu e reflexiva pe multimea '), write(A)), nl, tab(3),
        (sim(R), !, write('e simetrica'); write('nu e simetrica')), nl, tab(3),
        (tranz(R), !, write('e tranzitiva'); write('nu e tranzitiva')), nl.

/* Interogati:
?- A=[a,b,c,d,e,f], relP(P), relQ(Q), relR(R), relS(S), tiprel(P,A), tiprel(Q,A),
tiprel(R,A), tiprel(S,A).
?- relT(T), tiprel(T,[a,b,c,d]).
?- A=[a,b,c,d,e], relU(U), relV(V), tiprel(U,A), tiprel(V,A).
Relatia binara U={(a,a),(a,b),(b,a),(b,b),(c,c),(c,d),(d,c),(d,d),(e,e)} pe multimea
{a,b,c,d,e} (de cardinal 5, i.e. cu a,b,c,d,e doua cate doua distincte) este reflexiva,
simetrica si tranzitiva, asadar este (relatie de) echivalenta pe {a,b,c,d,e}, iar
V={(a,a),(a,b),(a,c),(b,b),(b,c),(c,c),(d,d),(d,e),(e,e)} este reflexiva si tranzitiva,
asadar este (relatie de) preordine pe aceasta multime. */

preord(R,A) :- refl(R,A), tranz(R).

eq(R,A) :- preord(R,A), sim(R).

/* Interogati:
?- relP(P), preord(P,[a,b,c,d,e]).
?- relV(V), preord(V,[a,b,c,d,e]).
*/

releq(R,A) :- relbinara(R,A), eq(R,A).
```

```
% Multimea relatiilor de echivalenta pe o multime A:

relatiieq(A,EqA) :- setof(R, releq(R,A), EqA).

/* Interogati:
?- relatiieq([a,b,c],L), afislista(L), length(L,N).
?- relatiieq([a,b,c,d],L), afislista(L), length(L,N).
*/

% Clasa C a unui element X raportat la o relatie de echivalenta E:

clasa(X,E,C) :- setof(Y, member((X,Y),E), C).

% Partitia P asociata unei relatii de echivalenta E pe o multime A:

parteq(E,A,P) :- setof(C, X^(member(X,A), clasa(X,E,C)), P).

% Variante recursive, fara setof, pentru cele doua predicate anterioare:

clasaelem(_,[],[]).
clasaelem(H,[(H,K)|T],[K|L]) :- clasaelem(H,T,L), !.
clasaelem(H,[_|T],C) :- clasaelem(H,T,C).

partasoceq(_, [], []).
partasoceq(R, [H|T], [C|LC]) :- clasaelem(H,R,C), diferenta(T,C,D),
                                partasoceq(R, D, LC).
```

```
/* Partitiile unei multimi A, cu prima, respectiv a doua varianta de mai sus de determinare
a partitiei asociate unei relatii de echivalenta: */

partitii(A,PartA) :- setof(P, R^(releq(R,A), parteq(R,A,P)), PartA).

partitiile(A,PartA) :- setof(P, R^(releq(R,A), partasoceq(R,A,P)), PartA).

/* Interogati:
?- partitii([a,b,c],L), afislista(L), length(L,N).
?- partitiile([a,b,c],L), afislista(L), length(L,N).
?- partitii([a,b,c,d],L), afislista(L), length(L,N).
?- partitiile([a,b,c,d],L), afislista(L), length(L,N).
Putem afisa simultan relatiile de echivalenta pe o multime si partitiile asociate lor: */

releqsipart(A,EqPartA) :- setof((R,P), R^(releq(R,A), parteq(R,A,P)), EqPartA).

releqnpart(A,EqPartA) :- setof((R,P),R^(releq(R,A),partasoceq(R,A,P)),EqPartA).

afislistaeqpart([]).
afislistaeqpart([(R,P)|T]) :- write(R), write(' are partitia asociata '),
                              write(P), nl, afislistaeqpart(T).

/* Interogati:
?- releqsipart([a,b,c],L), afislistaeqpart(L), length(L,N).
?- releqnpart([a,b,c],L), afislistaeqpart(L), length(L,N).
?- releqsipart([a,b,c,d],L), afislistaeqpart(L), length(L,N).
?- releqnpart([a,b,c,d],L), afislistaeqpart(L), length(L,N).
```

```
*/

% Relatia de echivalenta asociata unei partitii:

eqpart([],[]).
eqpart([A|LA],E) :- prodcart(A,A,AxA), eqpart(LA,F), reuniune(AxA,F,E).

/* Interogati:
?- eqpart([[a,b],[c]],R).
?- eqpart([[a,b],[c,d],[e]],U), write(U).
*/

/* Testarea antisimetriei, asimetriei, ireflexivitatii unei relatii binare pe o multime: */

antisim(R) :- not((member((X,Y),R), member((Y,X),R), X\=Y)).

asim(R) :- not((member((X,Y),R), member((Y,X),R))).

irefl(R) :- not(member((X,X),R)).

% Variante recursive, fara negatie, pentru cele trei predicate anterioare:

antisimetrica(R) :- auxantisim(R,R).

auxantisim(_,[]).
auxantisim(R,[(X,Y)|T]) :- (X=Y, !; not(member((Y,X),R))), auxantisim(R,T).
```

```
asimetrica(R) :- auxasim(R,R).

auxasim(_,[]).
auxasim(R,[(X,Y)|T]) :- not(member((Y,X),R)), auxasim(R,T).

ireflexiva([]).
ireflexiva([(X,Y)|T]) :- X\=Y, ireflexiva(T).

/* Predicatele de mai sus functioneaza doar pentru relatii binare pe multimi date ca liste
de constante in Prolog. */

% Relatiile de ordine pe o multime A, in doua variante:

ord(R,A) :- preord(R,A), antisim(R).

relord(R,A) :- relbinara(R,A), ord(R,A).

relatiiord(A,OrdA) :- setof(R, relord(R,A), OrdA).

ordine(R,A) :- preord(R,A), antisimetrica(R).

relordine(R,A) :- relbinara(R,A), ordine(R,A).

relatiiordine(A,OrdA) :- setof(R, relordine(R,A), OrdA).

% Relatiile de ordine stricta pe o multime A, in patru variante:
```

```
ordstr(R) :- asim(R), tranz(R).

relordstr(R,A) :- relbinara(R,A), ordstr(R).

relatiiordstr(A,OrdA) :- setof(R, relordstr(R,A), OrdA).

ordstricta(R) :- irefl(R), tranz(R).

relordstricta(R,A) :- relbinara(R,A), ordstricta(R).

relatiiordstricta(A,OrdA) :- setof(R, relordstricta(R,A), OrdA).

ordinestr(R) :- asimetrica(R), tranz(R).

relordinestr(R,A) :- relbinara(R,A), ordinestr(R).

relatiiordinestr(A,OrdA) :- setof(R, relordinestr(R,A), OrdA).

ordinestricta(R) :- ireflexiva(R), tranz(R).

relordinestricta(R,A) :- relbinara(R,A), ordinestricta(R).

relatiiordinestricta(A,OrdA) :- setof(R, relordinestricta(R,A), OrdA).

/* Interogati:
?- relatiiord([a,b,c],L), afislista(L), length(L,N).
?- relatiiordine([a,b,c],L), afislista(L), length(L,N).
```

```
?- relatiiordstr([a,b,c],L), afislista(L), length(L,N).
?- relatiiordstricta([a,b,c],L), afislista(L), length(L,N).
?- relatiiordinestr([a,b,c],L), afislista(L), length(L,N).
?- relatiiordinestricta([a,b,c],L), afislista(L), length(L,N).
*/

% Ordinea stricta S asociata unei ordini O, in doua moduri:

ordstrdinord(O,S) :- setof((X,Y), (member((X,Y), O), X\=Y), S), !.
ordstrdinord(_,[]).

ordstrictadinord([],[]).
ordstrictadinord([(X,X)|T],S) :- !, ordstrictadinord(T,S).
ordstrictadinord([(X,Y)|T],[(X,Y)|S]) :- ordstrictadinord(T,S).

/*  Alta varianta: includem ca argument multimea A pe care sunt definite S si O si sa
obtinem S ca diferenta de multimi intre O si diagonala lui A:*/

ordinestrictadinordine(O,A,S) :- diag(A,D), diferenta(O,D,S).

% Ordinea O asociata unei ordini stricte S pe o multime A:

orddinordstr(S,A,O) :- diag(A,D), reuniune(S,D,O).

/* Interogati:
?- orddinordstr([(a,b),(b,c),(a,c)],[a,b,c],O), ordstrdinord(O,S), ordstrictadinord(O,Str),
ordinestrictadinordine(O,[a,b,c],Stricta).
```

```
?- orddinordstr([(0,a),(0,b),(a,1),(b,1),(0,1)],[0,a,b,1],O), ordstrdinord(O,S),
ordstrictadinord(O,Str), ordinestrictadinordine(O,[0,a,b,1],Stricta).
*/

% Relatia de succesiune Succ asociata unei ordini stricte S:

succdinordstr(S,Succ) :- setof((X,Y), (member((X,Y),S),
                not((member((X,U),S), member((U,Y),S)))), Succ), !.
succdinordstr(_,[]).

% Relatia de succesiune Succ asociata unei ordini O:

succdinord(O,Succ) :- setof((X,Y), (member((X,Y),O), X\=Y,
        (not((member((X,U),O), X\=U, member((U,Y),O), U\=Y)))), Succ), !.
succdinord(_,[]).

/* Interogati:
?- S=[(a,b),(b,c),(a,c)], orddinordstr([(a,b),(b,c),(a,c)],[a,b,c],O),
succdinordstr(S,Succ1), succdinord(O,Succ2).
?- setof(Succ, R^(relord(R,[a,b,c]), succdinord(R,Succ)), L), afislista(L), length(L,N).
Cu interogarea anterioara am determinat toate relatiile de succesiune pe multimea {a,b,c}
(de cardinal 3, i.e. cu a,b,c doua cate doua distincte). */

%%%%%%%%%%%%%%%%%%%%%%%%%MATERIAL FACULTATIV:%%%%%%%%%%%%%%%%%%%%%%%%%
% Quicksort:

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

```
partitionare(P,[H|T],[H|S],D) :- H=<P, !, partitionare(P,T,S,D).
partitionare(P,[H|T],S,[H|D]) :- partitionare(P,T,S,D).

qsort([],[]).
qsort([H|T],L) :- partitionare(H,T,S,D), qsort(S,LS), qsort(D,LD),
                        append(LS,[H|LD],L).

% Sortarea rapida dupa un criteriu arbitrar:

partitionare(_,_,[],[],[]).
partitionare(Criteriu,P,[H|T],[H|S],D) :- Termen=..[Criteriu,H,P], Termen, !,
                partitionare(Criteriu,P,T,S,D).
partitionare(Criteriu,P,[H|T],S,[H|D]) :- partitionare(Criteriu,P,T,S,D).

qsort(_,[],[]).
qsort(Criteriu,[H|T],L) :- partitionare(Criteriu,H,T,S,D),
        qsort(Criteriu,S,LS), qsort(Criteriu,D,LD), append(LS,[H|LD],L).

/* Criteriile de sortare pot fi ordini sau preordini: de exemplu ordonarea listelor dupa
lungimea lor: */

ordlung(L,M) :- length(L,N), length(M,K), N=<K.

/* Interogati:
?- qsort([0,2,-1,5,7,5,0,1],L).
?- qsort(>=,[0,2,-1,5,7,5,0,1],L).
?- qsort(ordlung,[[a,b,c],[],[1,2],[X],[],[x,y]],L).
```

```
?- setof(Succ, R^(relord(R,[a,b,c]), succdinord(Succ,R)), L), qsort(ordlung,L,LS),
afislista(LS), length(LS,N).
Cu interogarea anterioara am determinat toate relatiile de succesiune pe multimea {a,b,c},
ordonate crescator dupa cardinal, i.e. dupa numarul de perechi pe care le contin.
Sa afisam relatiile de ordine stricta pe multimea {a,b,c}, iar ordonarea sa o efectuam dupa
cardinalul relatiilor de ordine stricta:
?- setof(S, R^(relord(R,[a,b,c]), ordstrdinord(S,R)), L), qsort(ordlung,L,LS),
afislista(LS), length(LS,N).
Sa afisam simultan ordinile pe multimea {a,b,c} si ordinile stricte si relatiile de
succesiune asociate acestora, ordonate dupa cardinalul ordinilor stricte:
?- setof((O,S,Succ), (relord(O,[a,b,c]), ordstrdinord(S,O), succdinordstr(Succ,S)), L),
qsort(ordlungmij,L,LS), afislistatripl(LS), length(LS,N). */

afislistatripl([]).
afislistatripl([(O,S,Succ)|T]) :- write(Succ),
        write(', ordinea stricta '), write(S), write(', ordinea '), write(O),
        nl, afislistatripl(T).

ordlungmij((_,L,_),(_,M,_)) :- ordlung(L,M).

/* Pentru a obtine relatiile de echivalenta pe multimea {a,b,c,d} si partitiile asociate
acestora, ordonate dupa cardinalele relatiilor de echivalenta, interogati:
?- releqnpart([a,b,c,d],L), qsort(ordlungprim,L,LS), afislistaeqpart(LS), length(LS,N). */

ordlungprim((L,_),(M,_)) :- ordlung(L,M).

/* Pentru a obtine partitiile multimii {a,b,c,d} ordonate ca in interogarea de mai sus,
```

```
interogati ca mai jos, cu lista partitiilor ordonata simultan descrescator dupa lungimea
fiecarei partitii si crescator dupa cardinalul maxim al unei clase a partitiei (pentru
multimi de cardinal >4 acest criteriu nu mai corespunde ordinii dupa cardinal intre
relatiile de echivalenta corespunzatoare):
?- partitiile([a,b,c,d],P), qsort(ord2lung,P,S), afislista(S), length(S,N).
*/

ordinvlung(L,M) :- length(L,N), length(M,K), N>=K.

ord2lung(L,M) :- ordinvlung(L,M), qsort(ordinvlung,L,[H|_]),
                 qsort(ordinvlung,M,[K|_]), ordlung(H,K).

/* Sau ca mai jos, cu lista partitiilor ordonata mai intai descrescator dupa lungimea
fiecarei partitii, apoi crescator dupa cardinalul maxim al unei clase a partitiei (pentru
multimi de cardinal >4 acest criteriu nu mai corespunde ordinii dupa cardinal intre
relatiile de echivalenta corespunzatoare):
?- partitiile([a,b,c,d],P), qsort(ordsucclung,P,S), afislista(S), length(S,N).
*/

ordsucclung(L,M) :- ordinvlung(L,M), (ordlung(L,M), !,
   qsort(ordinvlung,L,[H|_]), qsort(ordinvlung,M,[K|_]), ordlung(H,K); true).

% Maximul dupa lungime dintr-o lista de liste:

maximlung(A,B,A) :- ordlung(B,A), !.
maximlung(_,B,B).
```

```
maxlunglista([X],X).
maxlunglista([H|T],M) :- maxlunglista(T,MT), maximlung(H,MT,M).

% Varianta de scriere pentru ordinile anterioare:

ordmaxlung(L,M) :- maxlunglista(L,MaxL), maxlunglista(M,MaxM),
                        ordlung(MaxL,MaxM).

ord2lungv(L,M) :- ordlung(M,L), ordmaxlung(L,M).

ordsucclungv(L,M) :- ordlung(M,L), (ordlung(L,M), !, ordmaxlung(L,M); true).

/* Pentru a vedea diferenta dintre ordonarea ord2lung simultana dupa cele doua criterii si
ordonarea ordsucclung succesiva dupa cele doua criterii de mai sus, interogati:
?- ordsucclung([[],[1],[],[1,2,3,4,5]], [[a,b,c],[a,b]]).
?- ordsucclungv([[],[1],[],[1,2,3,4,5]], [[a,b,c],[a,b]]).
?- ord2lung([[],[1],[],[1,2,3,4,5]], [[a,b,c],[a,b]]).
?- ord2lungv([[],[1],[],[1,2,3,4,5]], [[a,b,c],[a,b]]).
*/

/* Ordonarea listelor de numere simultan crescator dupa lungime si dupa suma elementelor: */

suma([],0).
suma([H|T],S) :- suma(T,Scoada), S is H+Scoada.

ordsuma(L,M) :- suma(L,SL), suma(M,SM), SL=<SM.
```

```
ord2crit(L,M) :- ordlung(L,M), ordsuma(L,M).

/* Ordonarea listelor de numere crescator dupa lungime, apoi dupa suma elementelor: */

ordsuccrit(L,M) :- ordlung(L,M), (ordlung(M,L), !, ordsuma(L,M); true).

% Varianta de scriere:

ordtotsuccrit(L,M) :- ordlung(L,M), ordlung(M,L), !, ordsuma(L,M).
ordtotsuccrit(L,M) :- ordlung(L,M).

listaliste([[1,2,3,4,5],[0,1,0,3],[10,20,20],[100,2,30,4,5],[0,1,0,0]]).

/* Interogati:
?- ord2crit([1,2,3,4,5],[100,2,30,4,5]).
?- ord2crit([0,1,0,3],[100,2,30,4,5]).
?- ord2crit([10,20,20],[1,2,3,4,5]).
?- ordsuccrit([10,20,20],[1,2,3,4,5]).
?- ordtotsuccrit([10,20,20],[1,2,3,4,5]).
?- listaliste(L), qsort(ord2crit,L,S), afislista(S).
?- listaliste(L), qsort(ordsuccrit,L,S), afislista(S).
?- listaliste(L), qsort(ordtotsuccrit,L,S), afislista(S).
Acum sa ordonam liste de liste crescator mai intai dupa lungime, apoi dupa suma elementelor,
apoi dupa cel mai mare element din fiecare lista:
?- listal(L), qsort(ordsucc3crit,L,S), afislista(S).
?- altalistal(L), qsort(ordsucc3crit,L,S), afislista(S).
?- altall(L), qsort(ordsucc3crit,L,S), afislista(S). */
```

```
maxim(A,B,A) :- A>=B, !.
maxim(_,B,B).

maxlista([X],X).
maxlista([H|T],M) :- maxlista(T,MT), maxim(H,MT,M).

ordmax([],_) :- !.
ordmax(_,[]) :- fail.
ordmax(L,M) :- maxlista(L,MaxL), maxlista(M,MaxM), MaxL=<MaxM.

ordsucc3crit(L,M) :- ordlung(L,M), not(ordlung(M,L)).
ordsucc3crit(L,M) :- ordlung(L,M), ordlung(M,L),
                     ordsuma(L,M), not(ordsuma(M,L)).
ordsucc3crit(L,M) :- ordlung(L,M), ordlung(M,L),
                     ordsuma(L,M), ordsuma(M,L),
                     ordmax(L,M).

listal([[1,2,3,4,50],[0,1,0,3],[10,20,20],[10,2,30,40,5],[0,1,0,0],[10,20],[25,5]]).
altalistal([[40,15,5],[10,20,30],[1,50,2]]).
altall([[40,15,5],[1,2,3,4,50],[0,1,0,3],[10,20,20],[10,2,30,40,5],[10,20,30],[1,50,2],[0,1,
0,0],[10,20],[25,5]]).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Inchiderea reflexiva a unei relatii binare Q pe o multime A:
```

```
inchrefl(Q,A,R) :- diag(A,D), reuniune(D,Q,R).

% Inchiderea simetrica a unei relatii binare Q:

inchsim(Q,S) :- invrel(Q,R), reuniune(Q,R,S).

% Inchiderea tranzitiva a unei relatii binare Q:

inchtranz(Q,T) :- auxit(Q,1,[],T).

auxit(Q,N,R,T) :- putere(Q,N,P), reuniune(R,P,S),
                  (tranz(S), !, T=S ; SN is N+1, auxit(Q,SN,S,T)).

/* Inchiderea tranzitiva a unei relatii binare Q, evitand calculele repetate ale puterilor:
*/

inchidtranz(Q,T) :- auxitr(Q,Q,T).

auxitr(Q,R,T) :- tranz(R), !, T=R ;
                 comprel(Q,R,S), reuniune(Q,S,P), auxitr(Q,P,T).

% Preordinea generata de o relatie binara Q pe o multime A:

preordgen(Q,A,P) :- inchidtranz(Q,T), inchrefl(T,A,P).

% Echivalenta generata de o relatie binara Q pe o multime A:
```

```
eqgen(Q,A,E) :- inchsim(Q,S), preordgen(S,A,E).

% Ordinea stricta S a unui poset finit avand relatia de succesiune Succ:

ordstrdinsucc(Succ,S) :- inchidtranz(Succ,S).

/* Ordinea O a unui poset finit avand multimea suport A si relatia de succesiune Succ: */

orddinsucc(Succ,A,O) :- preordgen(Succ,A,O).
```