```
are(ana,mere).
are(vali,mere).
are(bogdan,pere).
are(nicu,Fructe) :- are(ana,Fructe), are(bogdan,Fructe).
are(maria,Fructe) :- are(ana,Fructe); are(bogdan,Fructe).

/* Amintesc ca numele de variabile incep cu litera mare sau underscore. Asadar nu puteam
sa scriem cu prima litera mare constantele ana, vali, bogdan, nicu, maria sau constantele mere, pere
sau predicatul binar are, si nici variabila Fructe cu prima litera mica.
   Amintesc ca predicatul binar infixat "," este conjunctia, iar ";" este disjunctia logica.
Predicatele unare "\+" si "not" reprezinta negatia logica.
   Amintesc ca precedentele sunt opusul prioritatilor. Conjunctia are precedenta mai mica, deci
prioritatea mai mare decat a disjunctiei. Variabilele, constantele, termenii scrisi sub forma
nume_operator(argumentul1,...,argumentulN) si termenii incadrati intre paranteze au precedenta 0,
asadar au cea mai mare prioritate, intrucat precedentele sunt numere naturale.
   Interogati (peste tot marchez interogarile ca mai jos, precedandu-le de cursorul "?-" al
interpretorului Prolog-ului):
?- are(ana,mere).
?- are(ana,pere).
?- are(ana,Ce).
?- are(nicu,Ce).
?- are(maria,Ce).
?- are(Cine,mere).
?- are(Cine,Ce).
si dati ; (in Prolog-ul desktop) / Next (sau 10/100 etc. in Prolog-ul online) pentru a
obtine toate solutiile ultimelor doua interogari. La final, Prolog-ul va intoarce false, indicand
faptul ca nu mai exista ale solutii.
   Dati si scopul compus:
?- (are(vali,Fructe); are(bogdan,Fructe)), are(ana,Fructe).
*/
```

```
/* Amintesc ca interogarile, la fel ca faptele si regulile, sunt predicate, adica termeni cu
operatorii dominanti booleeni. Pentru calcule aritmetice vom interoga sub forma:
?- Cat is 2**10.
cu predicatul binar infixat is, care are ca efect calcularea expresiei aritmetice din
argumentul sau drept, urmata de unificarea valorii astfel obtinute cu argumentul sau stang.
Folosirea egalului de unificare in locul lui "is":
?- Cat=2**10.
duce la unificarea variabilei Cat cu expresia aritmetica:
   **
  /  \
 2   10
Prolog-ul efectueaza foarte rapid calcule aritmetice cu numere foarte mari. Interogati:
?- Cat is 125**107.
?- Cat is 12.5**10.7.
Amintesc ca, in Prolog-ul desktop, egalul de unificare nu testeaza si aparitia variabilelor in
termenii cu care dorim sa le unificam:
?- X=f(X).
Sigur ca unificarea de mai sus nu are loc, pentru ca ar conduce la unificarea lui X cu un termen
infinit. Pentru unificare cu testarea ocurentelor variabilelor in termeni interogam:
?- unify_with_occurs_check(X,f(X)).
*/

/* Amintesc ca, in scrierea predicatelor din documentatia Prolog-ului, argumentele precedate de "+"
trebuie furnizate Prolog-ului in interogari, cele precedate de "-" sunt calculate de Prolog in
interogari, iar cele precedate de "?" pot avea ambele roluri.
   De exemplu, urmatorul predicat pentru calculul lungimii unei liste functioneaza sub forma
lungime(?Lista,?NrElemente); vedeti si predicatul predefinit "length" care efectueaza acest lucru:
*/

lungime([],0).
lungime([_|T],N) :- lungime(T,K), N is K+1.
```

```
/* Interogati:
?- lungime([1,B,c],Cat).
?- lungime(CeFelDeLista,3).
*/

/* Calculul factorialului: factorial(+N,-Nfactorial), factorialDeCat(-N,+Nfactorial);
predicate echivalente cu acestea: factorialul(+N,-Nfactorial), factDeCat(-N,+Nfactorial), respectiv:
*/

factorial(0,1).
factorial(N,F) :- N>0, PN is N-1, factorial(PN,G), F is G*N.

/* Interogati:
?- factorial(5,Cat).
?- factorial(500,Cat).
*/

factorialul(0,1) :- !.
factorialul(N,F) :- N>0, PN is N-1, factorialul(PN,G), F is G*N.

/* Amintesc ca predicatul cut ("!") taie backtracking-ul executat de Prolog, astfel ca Prolog-ul nu
mai cauta alte satisfaceri ale scopului curent dupa ce intalneste acest predicat. Interogati:
?- factorialul(5,Cat).
?- factorialul(500,Cat).
*/

factorialDeCat(N,F) :- auxfactorial(N,F,0,1).

auxfactorial(N,F,N,F).
auxfactorial(N,F,K,G) :- G<F, SK is K+1, UG is G*SK, auxfactorial(N,F,SK,UG).
```

```
/* Interogati:
?- factorialDeCat(N,120).
?- factorialDeCat(N,100).
?- factorialDeCat(N,0).
?- factorialDeCat(N,1).
*/

factDeCat(0,1).
factDeCat(N,F) :- auxfact(N,F,1,1).

auxfact(N,F,N,F).
auxfact(N,F,K,G) :- G<F, SK is K+1, UG is G*SK, auxfact(N,F,SK,UG).

/* Interogati:
?- factDeCat(N,120).
?- factDeCat(N,100).
?- factDeCat(N,0).
?- factDeCat(N,1).
Observati ca aceasta varianta a predicatului da ambele solutii pentru ultima interogare.
*/

/* Interogati:
?- CatImpartireIntreaga is -10 div 3.
?- RestImpartireIntreaga is -10 mod 3.
?- AltRest is -10 rem 3.
?- ValoareaAbsoluta is abs(-10).
*/

% Predicat unar care determina daca argumentul sau e numar intreg prim:
```

```
prim(N) :- integer(N), V is abs(N), ePrim(V).

% Varianta:

nrPrim(N) :- integer(N), (N>=0, ePrim(N) ; N<0, V is -N, ePrim(V)).

% Sau, echivalent:

numarPrim(N) :- integer(N), (N>=0, !, ePrim(N) ; V is -N, ePrim(V)).

% Test cut:

primCuTestCut(N) :- integer(N), (N>=0, !, ePrim(N) ; write(oups), V is -N, ePrim(V)).

% nusediv(D,N)=true <=> N nu se divide cu niciunul dintre numerele D,D+1,...,N-1
ePrim(N) :- nusediv(2,N).

nusediv(N,N).
nusediv(D,N) :- D<N, N mod D>0, SD is D+1, nusediv(SD,N).

/* Interogati:
?- prim(11).
?- prim(-11).
?- prim(10).
?- nrPrim(11).
?- nrPrim(10).
?- numarPrim(11).
?- numarPrim(10).
?- primCuTestCut(11).
?- primCuTestCut(10).
```

```
Daca ar fi afisat "oups", ar fi insemnat ca, dupa e ePrim(10) esueaza, ar fi trecut de cut. Dar
predicatul cut taie backtracking-ul indiferent de rezultatul clauzelor care il urmeaza. Afisarea lui
"oups" se face la urmatoarele interogari:
?- primCuTestCut(-11).
?- primCuTestCut(-10).
*/

/* Lista numerelor naturale prime mai mici sau egale cu un numar natural N:
listaPrime(+N,-ListaPrime): */

listaPrime(N,[]) :- N<2, !.
listaPrime(N,[N|L]) :- ePrim(N), !, PN is N-1, listaPrime(PN,L).
listaPrime(N,L) :- PN is N-1, listaPrime(PN,L).

/* Interogati:
?- listaPrime(1000,L), write(L).
Listele lungi sunt date truchiat de Prolog ca valori de variabile, asa ca necesita afisare
explicita. */

/* Ciurul lui Eratostene pentru determinarea listei numerelor naturale prime mai mici sau egale cu
un numar natural N: ciur(+N,-ListaPrime): */

ciur(N,LP) :- lista(N,L), ciuruire(L,LP).

lista(N,L) :- auxlista(2,N,L).

auxlista(K,N,[]) :- K>N, !.
auxlista(K,N,[K|L]) :- SK is K+1, auxlista(SK,N,L).

ciuruire([],[]).
ciuruire([H|T],[H|L]) :- filtreaza(H,T,M), ciuruire(M,L).
```

```
filtreaza(_,[],[]).
filtreaza(X,[H|T],[H|L]) :- H mod X>0, !, filtreaza(X,T,L).
filtreaza(X,[_|T],L) :- filtreaza(X,T,L).

/* Interogati:
?- ciur(1000,L), write(L).
*/
```