```
:-dynamic pos/1, minim/1, minp/1.


/* ------------------------------------------------------------ */
/* 1 */

/* atLeast(PredList,ParamList,N) */

atLeast([],_,N) :-
  N>0, !, fail.
atLeast(_,_,N) :-
  N=<0, !.
atLeast([P|PS],AS,N) :-
  C =.. [P|AS],
  call(C), !,
  NN is N-1,
  atLeast(PS,AS,NN).
atLeast([_|PS],AS,N) :-
  atLeast(PS,AS,N).

/* --- na otestovani --- */
gt(X,Y) :- X>Y.
ge(X,Y) :- X>=Y.
lt(X,Y) :- X<Y.
le(X,Y) :- X=<Y.
eq(X,Y) :- X==Y.
ne(X,Y) :- X\==Y.

/* ------------------------------------------------------------ */
/* 2 */

/*

lvar(NAME).     promenna
lapp(E1,E2).    aplikace
labs(NAME,E).   abstrakce

*/

/* fv(Where,FreeVars) */


fv(X,Y) :-
  fv(X,[],Y).

fv(lvar(N),B,[]) :-
  member(N,B),!.
fv(lvar(N),_,[N]).
fv(lapp(E1,E2),B,R) :-
  fv(E1,B,R1),
  fv(E2,B,R2),
  uni(R1,R2,R).
fv(labs(N,E),B,R) :-
  fv(E,[N|B],R).

uni([],X,X) :- !.
uni(X,[],X) :- !.
uni([H|T],X,Y) :-
  member(H,X),!,
  uni(T,X,Y).
uni([H|T],X,[H|Y]) :-
  uni(T,X,Y).


/* ------------------------------------------------------------ */
/* 3 */

midLen([],_) :- !, fail.
midLen([X|Y],R) :-
  length(X,L),
  getMML(Y,L,I,L,A),
  MID is (I+A) // 2,
  selM(Y,MID,X,L,R).


selM([],_,X,_,X).
selM([H|T],MID,X,L,R) :-
  length(H,LH),
  LM is abs(L-MID),
  HM is abs(LH-MID),
  (HM <  LM, selM(T,MID,H,LH,R);
   HM >= LM, selM(T,MID,X,L,R)).


getMML([],I,I,A,A).
getMML([H|T],TI,I,TA,A) :-
  length(H,L),
  (L<TI,II=L;L>=TI,II=TI),
  (L>TA,AA=L;L=<TA,AA=TA),
  getMML(T,II,I,AA,A).


/* ------------------------------------------------------------ */
/* 4 */

search(From,To,Shrt) :-
  retractall(pos(_)),
  retractall(minim(_)),
  retractall(minp(_)),
  assertz(minim(-1)),
  assertz(minp([])),
  bagof(L,findp(From,To,0),_),
  minp(Shrt).

findp(F,F,X) :-
  (minim(-1),retract(minim(-1));
   minim(MV),X<MV,retract(minim(MV))),
  assertz(pos(F)),
  bagof(P,pos(P),MP),
  retract(minp(_)),
  assertz(minp(MP)),
  assertz(minim(X)).
findp(F,F,_) :-
  !, pos(F),
  retract(pos(F)),
  fail.
findp(F,T,N) :-
  assertz(pos(F)),
  nextStep(F,NF),
  not(pos(NF)),
  NN is N+1,
  findp(NF,T,NN).
findp(F,_,_) :-
  pos(F),
  retract(pos(F)),
  !, fail.


/* toto nebylo treba */

nextStep(p(X,Y),p(XX,YY)) :-
  move(I,J),
  XX is X+I,
  YY is Y+J,
  XX > 0, YY > 0,
  XX < 6, YY < 6.

move(1,0).
move(0,1).
move(-1,0).
move(0,-1).


/* ------------------------------------------------------------ */
/* ------------------------------------------------------------ */
```