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

:- dynamic
    pom/1,
    price/1,
    ppos/1.

/* ------------------------------------------------------------ */
/* ------------------------------------------------------------ */

getPath(S,E,Path) :-
  retractall(pom(_)),
  retractall(price(_)),
  retractall(ppos(_)),
  assert(price(none)),
  mks(S,E,Path).

mks(S,E,_) :-
  search(S,E,0,[]).
mks(S,_,[S|Path]) :-
  ppos(Path).

checkP(NP,Path) :-
  price(none),
  retract(price(none)),
  assert(price(NP)),
  assert(ppos(Path)),
  !,fail.
checkP(NP,Path) :-
  price(P),
  NP<P,
  retract(price(P)),
  retract(ppos(_)),
  assert(price(NP)),
  assert(ppos(Path)),
  !,fail.

search(E,E,Price,Path) :- !, checkP(Price,Path).
search(S,E,P,TP) :-
  assertz(pom(S)),
  nextStep(S,Nxt,SP),
  not(pom(Nxt)),
  NP is P+SP,
  append(TP,[Nxt],NTP),
  search(Nxt,E,NP,NTP).
search(S,_,_,_) :-
  pom(S),
  retract(pom(S)),
  !, fail.


/* ------------ */
/* jen pro test */


nextStep(pos(X,Y),pos(XX,Y),P) :-
  XX is X + 1, check(XX,Y), P is (XX-3)*(XX-3)+(Y-3)*(Y-3).
nextStep(pos(X,Y),pos(X,YY),P) :-
  YY is Y + 1, check(X,YY), P is (X-3)*(X-3)+(YY-3)*(YY-3).
nextStep(pos(X,Y),pos(XX,Y),P) :-
  XX is X - 1, check(XX,Y), P is (XX-3)*(XX-3)+(Y-3)*(Y-3).
nextStep(pos(X,Y),pos(X,YY),P) :-
  YY is Y - 1, check(X,YY), P is (X-3)*(X-3)+(YY-3)*(YY-3).

check(X,Y) :-
  X > 0, X =< 5, Y > 0, Y =< 5.


/* ------------------------------------------------------------ */
/* ------------------------------------------------------------ */

/* or(L,R) and(L,R) not(E) true false var(V) */
/* [w(Var,Value)] */
```

```
eval(_,true,true).
eval(_,false,false).
eval(T,var(V),R) :-
    getVal(T,V,R),!.
eval(T,not(E),R) :-
    eval(T,E,EE),
    (EE=true,R=false ;
     R=true ),!.
eval(T,and(E1,E2),R) :-
    eval(T,E1,EE1),
    (EE1=false,R=false ;
     eval(T,E2,R) ),!.
eval(T,or(E1,E2),R) :-
    eval(T,E1,EE1),
    (EE1=true,R=true ;
     eval(T,E2,R) ),!.


getVal([w(V,Value)|_],V,Value) :- !.
getVal([_|WS],V,Value) :-
    getVal(WS,V,Value).


/* ------------------------------------------------------------ */
/* ------------------------------------------------------------ */

merge([],L,L).
merge(L,[],L).
merge([H1|T1],[H2|T2],[H1|TT]) :-
    H1 =< H2,
    merge(T1,[H2|T2],TT).
merge([H1|T1],[H2|T2],[H2|TT]) :-
    H2 < H1,
    merge([H1|T1],T2,TT).

msort([],[]).
msort([V],[V]).
msort([A,B|T],R) :-
    divide(T,L1,L2),
    msort([A|L1],S1),
    msort([B|L2],S2),
    merge(S1,S2,R).

divide([],[],[]).
divide([V],[V],[]).
divide([A,B|T],[A|TA],[B|TB]) :-
    divide(T,TA,TB).


/* ------------------------------------------------------------ */
/* ------------------------------------------------------------ */

mapC(_,[],[]) :- !.
mapC(F,[HL|TL],RES) :-
    mapC(F,TL,RT),
    mapA(F,HL,RT,RES).

mapA(_,[],R,R) :- !.
mapA(F,[H|T],R,[X|RES]) :-
    C =.. [F,H,X],
    call(C),
    mapA(F,T,R,RES).


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