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

:-dynamic
    w/2.

pt(Xs,Ys,p(X,Y),p(XX,YY),N) :-
  retractall(w(_,_)),
  setof(Path,D^search(0,D,p(X,Y),p(XX,YY),Path),Ps),
  length(Ps,N).

search(3,_,p(X,Y),p(X,Y),[p(X,Y)]) :- !.
search(N,D,p(X,Y),p(XX,YY),[p(X,Y) | R]) :-
  assert(w(X,Y)),
  getNext(N,D,X,Y,NX,NY,ND,NN),
  NN =< 3,
  not(w(NX,NY)),
  search(NN,ND,p(NX,NY),p(XX,YY),R).
search(_,_,p(X,Y),_,_) :-
  retract(w(X,Y)),
  !,fail.

getNext(N,1,X,Y,NX,NY,1,N) :-
  X<8, NX is X+1, NY = Y.
getNext(N,D,X,Y,NX,NY,1,NN) :-
  nonvar(D), D\==1, X<8, NX is X+1, NY = Y, NN is N+1.

/* to nize stacilo jen lehce naznacit */

getNext(N,3,X,Y,NX,NY,3,N) :-
  NX is X-1, NX>0, NY = Y.
getNext(N,D,X,Y,NX,NY,3,NN) :-
  nonvar(D), D\==3, NX is X-1, NX>0, NY = Y, NN is N+1.

getNext(N,2,X,Y,NX,NY,2,N) :-
  Y<8, NY is Y+1, NX = X.
getNext(N,D,X,Y,NX,NY,2,NN) :-
  nonvar(D), D\==2, Y<8, NY is Y+1, NX = X, NN is N+1.

getNext(N,4,X,Y,NX,NY,4,N) :-
  NY is Y-1, NY>0, NX = X.
getNext(N,D,X,Y,NX,NY,4,NN) :-
  nonvar(D), D\==4, NY is Y-1, NY>0, NX = X, NN is N+1.


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

/* toto je jen ukazka, na vysvetlenou */
tree1( node(2,node(1,lf,lf),node(3,lf,lf)) ).
tree2( node(4,node(3,lf,lf),node(5,lf,lf)) ).
tree3( node(4,node(2,node(1,lf,lf),node(3,lf,lf)), node(5,lf,lf)) ).
/* toto je jen ukazka, na vysvetlenou */

/* jen testovaci predikat */
odd(X) :- Y is (X // 2) * 2, Y \== X.


/* tu zacina vlstni reseni */
eqSub(X,X) :- !.
eqSub(node(_,L,R),X) :-
  eqSub(L,X),!
  ;
  eqSub(R,X).

subs(lf,_,_,lf).
subs(node(Val,L,R), P, V, node(V,NL,NR)) :-
  PP =.. [P,Val], call(PP), !,
  subs(L,P,V,NL), subs(R,P,V,NR).
subs(node(Val,L,R), P, V, node(Val,NL,NR)) :-
  subs(L,P,V,NL), subs(R,P,V,NR).

subsIFsubt(Sub,Tree, P, V, NT) :-
  eqSub(Tree,Sub), !, subs(Tree, P, V, NT).
```

```
subsIFsubt(_  ,Tree, _, _, Tree).

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

/* include je filter, co jste meli preddefinovany */
allpal(S,R) :-
  suff(S,SS), pref(SS,L), include(len,L,FL), include(pal,FL,R).

len(L) :- length(L,LL), LL>2.
pal(S) :- reverse(S,S).

suff([],[]).
suff([X|XS], [[X|XS] | R]) :- suff(XS, R).

pref([S|SS], Res) :-
  reverse(S,RS),suff(RS,RevSS),
  mapRev(RevSS,PS), pref(SS,PXS),
  append(PS,PXS,Res).
pref([],[]).

mapRev([],[]).
mapRev([X|XS],[XR|XRS]) :- reverse(X,XR), mapRev(XS,XRS).

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

union([],X,X).
union(X,[],X):-!.
union([X|XS],Y,R) :-
  mmber(X,Y),!,union(XS,Y,R).
union([_|XS],Y,R) :-
  union(XS,Y,R).

mmber(X,[X|_]) :- !.
mmber(X,[_|XS]) :- mmber(X,XS).

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