```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%      House Reconfiguration Problem      %%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% domain sets
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% legacy - only short things
thing(T) :- legacyConfig(thing(T)).
% new - short and long
thing(T) :- thingLong(T).
thing(T) :- thingShort(T).
% a thing can't be both
thingShort(T) :- thing(T), not thingLong(T).
:- thingLong(T), thingShort(T).

% all possible cabinets
cabinetDomain(C) :- legacyConfig(cabinet(C)).
cabinetDomain(C) :- cabinetDomainNew(C).

% all possible rooms
roomDomain(R) :- legacyConfig(room(R)).
roomDomain(R) :- roomDomainNew(R).

person(P) :- legacyConfig(person(P)).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% solution mappings
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% each thing is owned by a single person
personTOthing(P, T) :- legacyConfig(personTOthing(P, T)).

% each thing is stored in a single cabinet
1 { cabinetTOthing(C, T) : cabinetDomain(C) } 1 :- thing(T).

% each cabinet is located in a single room
1 { roomTOcabinet(R, C) : roomDomain(R) } 1 :- cabinet(C).

% each room belongs to a single person, who owns things stored in cabinets that
are located in that room
personTOroom(P, R) :- personTOthing(P, T), cabinetTOthing(C, T), roomTOcabinet(R,
C).
```

```
legacyConfig(personTOroom(P, R)) :- legacyConfig(personTOthing(P, T)),
legacyConfig(cabinetTOthing(C, T)), legacyConfig(roomTOcabinet(R, C)).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% solution sets
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% cabinet
cabinet(C) :- cabinetTOthing(C, T).
cabinet(C) :- roomTOcabinet(R, C).
% subtypes
1 { cabinetSmall(C); cabinetHigh(C) } 1 :- cabinet(C).
% long things can only be stored in high cabinets
cabinetHigh(C) :- thingLong(T), cabinetTOthing(C, T).

% room
room(R) :- roomTOcabinet(R, C).
room(R) :- personTOroom(P, R).

% consistent ordering for symmetry breaking
cabinet(C1) :- cabinetDomainNew(C1), cabinetDomainNew(C2), cabinet(C2), C1 < C2.
room(R1) :- roomDomainNew(R1), roomDomainNew(R2), room(R2), R1 < R2.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% constraints
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% no shared ownership of things or rooms
:- personTOthing(P1, T), personTOthing(P2, T), P1 != P2.
:- personTOroom(P1, R), personTOroom(P2, R), P1 != P2.

% no multi-tenancy in the same cabinet either
:- cabinetTOthing(C, T1), cabinetTOthing(C, T2), personTOthing(P1, T1),
personTOthing(P2, T2), P1 != P2.

% a cabinet can contain at most 5 things
:- 6 { cabinetTOthing(C, T) : thing(T) }, cabinet(C).
% a room can contain at most 4 cabinets
:- 5 { roomTOcabinet(R, C) : cabinetDomain(C)}, room(R).

% high cabinets take 2 slots, small cabinets take 1 slot
cabinetSize(C, 1) :- cabinetSmall(C).
cabinetSize(C, 2) :- cabinetHigh(C).
roomSlotUsage(R, C, S) :- roomTOcabinet(R, C), cabinetSize(C, S).
:- #sum { S, C : roomSlotUsage(R, C, S) } > 4, room(R).
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% reconfiguration
%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% legacy -> solution === either reuse or remove elements
1 { reuse(room(R)); remove(room(R)) } 1 :- legacyConfig(room(R)).
1 { reuse(cabinet(C)); remove(cabinet(C)) } 1 :- legacyConfig(cabinet(C)).
1 { reuse(personTOroom(P, R)); remove(personTOroom(P, R)) } 1 :-
legacyConfig(personTOroom(P, R)).
1 { reuse(roomTOcabinet(R, C)); remove(roomTOcabinet(R, C)) } 1 :-
legacyConfig(roomTOcabinet(R, C)).
1 { reuse(cabinetTOthing(C, T)); remove(cabinetTOthing(C, T)) } 1 :-
legacyConfig(cabinetTOthing(C, T)).

% reused elements MUST be a part of the solution
room(R) :- reuse(room(R)).
cabinet(C) :- reuse(cabinet(C)).
personTOroom(P, R) :- reuse(personTOroom(P, R)).
roomTOcabinet(R, C) :- reuse(roomTOcabinet(R, C)).
cabinetTOthing(C, T) :- reuse(cabinetTOthing(C, T)).

% deleted elements CANNOT be a part of the solution
:- room(R), remove(room(R)).
:- cabinet(C), remove(cabinet(C)).
:- personTOroom(P, R), remove(personTOroom(P, R)).
:- roomTOcabinet(R, C), remove(roomTOcabinet(R, C)).
:- cabinetTOthing(C, T), remove(cabinetTOthing(C, T)).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% cost defaults
%%%%%%%%%%%%%%%%%%%%%%%%%%%%

cabinetTOthingCost(0).
finalcabinetTOthingCost(W) :- cabinetTOthingCost(0), #max {X, 1 :
cabinetTOthingCost(X)} = W.
roomTOcabinetCost(0).
finalroomTOcabinetCost(W) :- roomTOcabinetCost(0), #max {X, 1 :
roomTOcabinetCost(X)} = W.
personTOroomCost(0).
finalpersonTOroomCost(W) :- personTOroomCost(0), #max {X, 1 :
personTOroomCost(X)} = W.
cabinetHighCost(0).
finalcabinetHighCost(W) :- cabinetHighCost(0), #max {X, 1 : cabinetHighCost(X)} =
W.
```

```
cabinetSmallCost(0).
finalcabinetSmallCost(W) :- cabinetSmallCost(0), #max {X, 1 :
cabinetSmallCost(X)} = W.
roomCost(0).
finalroomCost(W) :- roomCost(0), #max {X, 1 : roomCost(X)} = W.
reuseCabinetTOthingCost(0).
finalreuseCabinetTOthingCost(W) :- reuseCabinetTOthingCost(0), #max {X, 1 :
reuseCabinetTOthingCost(X)} = W.
reuseRoomTOcabinetCost(0).
finalreuseRoomTOcabinetCost(W) :- reuseRoomTOcabinetCost(0), #max {X, 1 :
reuseRoomTOcabinetCost(X)} = W.
reusePersonTOroomCost(0).
finalreusePersonTOroomCost(W) :- reusePersonTOroomCost(0), #max {X, 1 :
reusePersonTOroomCost(X)} = W.
reuseCabinetAsSmallCost(0).
finalreuseCabinetAsSmallCost(W) :- reuseCabinetAsSmallCost(0), #max {X, 1 :
reuseCabinetAsSmallCost(X)} = W.
reuseCabinetAsHighCost(0).
finalreuseCabinetAsHighCost(W) :- reuseCabinetAsHighCost(0), #max {X, 1 :
reuseCabinetAsHighCost(X)} = W.
reuseRoomCost(0).
finalreuseRoomCost(W) :- reuseRoomCost(0), #max {X, 1 : reuseRoomCost(X)} = W.
removeCabinetTOthingCost(0).
finalremoveCabinetTOthingCost(W) :- removeCabinetTOthingCost(0), #max {X, 1 :
removeCabinetTOthingCost(X)} = W.
removeRoomTOcabinetCost(0).
finalremoveRoomTOcabinetCost(W) :- removeRoomTOcabinetCost(0), #max {X, 1 :
removeRoomTOcabinetCost(X)} = W.
removePersonTOroomCost(0).
finalremovePersonTOroomCost(W) :- removePersonTOroomCost(0), #max {X, 1 :
removePersonTOroomCost(X)} = W.
removeCabinetCost(0).
finalremoveCabinetCost(W) :- removeCabinetCost(0), #max {X, 1 :
removeCabinetCost(X)} = W.
removeRoomCost(0).
finalremoveRoomCost(W) :- removeRoomCost(0), #max {X, 1 : removeRoomCost(X)} = W.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% cost assignment
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% reusing
cost(reuse(cabinetTOthing(C, T)), W) :- cabinetTOthing(C, T),
legacyConfig(cabinetTOthing(C, T)), finalreuseCabinetTOthingCost(W).
```

```
cost(reuse(roomTOcabinet(R, C)), W) :- roomTOcabinet(R, C),
legacyConfig(roomTOcabinet(R, C)), finalreuseRoomTOcabinetCost(W).
cost(reuse(personTOroom(P, R)), W) :- personTOroom(P, R),
legacyConfig(personTOroom(P, R)), finalreusePersonTOroomCost(W).
cost(reuse(cabinetHigh(C)), W) :- cabinetHigh(C), legacyConfig(cabinet(C)),
finalreuseCabinetAsHighCost(W).
cost(reuse(cabinetSmall(C)), W) :- cabinetSmall(C), legacyConfig(cabinet(C)),
finalreuseCabinetAsSmallCost(W).
cost(reuse(room(R)), W) :- room(R), legacyConfig(room(R)), finalreuseRoomCost(W).

% removing
cost(remove(cabinetTOthing(C, T)), W) :- not cabinetTOthing(C, T),
legacyConfig(cabinetTOthing(C, T)), finalremoveCabinetTOthingCost(W).
cost(remove(roomTOcabinet(R, C)), W) :- not roomTOcabinet(R, C),
legacyConfig(roomTOcabinet(R, C)), finalremoveRoomTOcabinetCost(W).
cost(remove(personTOroom(P, R)), W) :- not personTOroom(P, R),
legacyConfig(personTOroom(P, R)), finalremovePersonTOroomCost(W).
cost(remove(cabinet(C)), W) :- not cabinet(C), legacyConfig(cabinet(C)),
finalremoveCabinetCost(W).
cost(remove(room(R)), W) :- not room(R), legacyConfig(room(R)),
finalremoveRoomCost(W).

% creating
cost(create(cabinetHigh(C)), W) :- cabinet(C), not legacyConfig(cabinet(C)),
cabinetHigh(C), finalcabinetHighCost(W).
cost(create(cabinetSmall(C)), W) :- cabinet(C), not legacyConfig(cabinet(C)),
cabinetSmall(C), finalcabinetSmallCost(W).
cost(create(room(R)), W) :- room(R), not legacyConfig(room(R)), finalroomCost(W).
cost(create(personTOroom(P, R)), W) :- personTOroom(P, R), not
legacyConfig(personTOroom(P, R)), finalpersonTOroomCost(W).
cost(create(roomTOcabinet(R, C)), W) :- roomTOcabinet(R, C), not
legacyConfig(roomTOcabinet(R, C)), finalroomTOcabinetCost(W).
cost(create(cabinetTOthing(C, T)), W) :- cabinetTOthing(C, T), not
legacyConfig(cabinetTOthing(C, T)), finalcabinetTOthingCost(W).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% optimization
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

#minimize { W, X : cost(X, W) }.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% output
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
#show cabinet/1.
#show cabinetHigh/1.
#show cabinetSmall/1.
#show room/1.
#show cabinetTOthing/2.
#show roomTOcabinet/2.
#show personTOroom/2.
% #show cost/2.
```