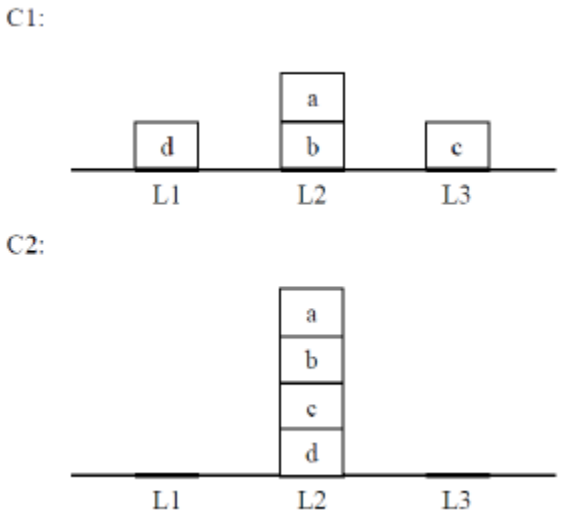
# Laborator 7

- Problema mutarii blocurilor, folosind algoritmii de cautare neinformata breadth first, dept first si depth first  iterative deepening.

Sa consideram ca avem la dispozitie M blocuri, depozitate pe un teren. Pe suprafata terenului exista un numar de N locatii de depozitare. Pentru o mai buna utilizare a spatiului, blocurile pot fi asezate unele peste altele, in stive. In fiecare locatie de depozitare exista cate o astfel de stiva, eventual vida. Un bloc poate fi mutat din locul sau numai daca el se afla in varful unei stive, si poate fi asezat numai deasupra unei alte stive (eventual vide).

Pentru doua configuratii date C1, C2 ale asezarii celor M blocuri in cele N locatii, sa se tabileasca daca si cum este posibil sa se ajunga in configuratia C2, plecand din configuratia C1.

Exemplu pentru M = 4 si N = 3:




In Prolog, reprezentam o stare a problemei (o configuratie) prin intermediul unei liste de liste C. Fiecare lista din C corespunde unei stive de blocuri si este ordonata astfel incat blocul din varful stivei corespunde capului listei. Stivele vide sunt reprezentate prin liste vide. In cazul exemplului anterior, termenii Prolog corespunzatori lui C1, respectiv C2 sunt: [[d],[a,b],[c]] si [[ ],[a,b,c,d],[ ]].

Pornind de la codul de mai jos, implementati predicatele marcate cu rosu. Explicatia fiecaruia dintre aceste predicate se gaseste pe ultima pagina.

```
%% START mutare blocuri
% ----------------------------------------
% Cautare de tip breadth-first
% ----------------------------------------
rezolva_b(NodInitial,Solutie):- breadthfirst([[NodInitial]],Solutie).

breadthfirst([[Nod|Drum]|_],[Nod|Drum]):-scop(Nod).
breadthfirst([Drum|Drumuri],Solutie):- extinde(Drum,DrumNoi),
                                       append(Drumuri,DrumNoi,Drumuri1),
                                       breadthfirst(Drumuri1,Solutie).

extinde([Nod|Drum],DrumNoi):-
                bagof([NodNou,Nod|Drum],(s(Nod,NodNou),
                        \+ (member(NodNou,[Nod|Drum]))), DrumNoi),!.
extinde(_,[]).
% ----------------------------------------
%Cautare de tip depth-first cu mecanism de detectare a ciclurilor
% ----------------------------------------
rezolva(Nod,Solutie):-depthfirst([],Nod,Solutie).

depthfirst(Drum, Nod,[Nod|Drum]):-scop(Nod).
depthfirst(Drum,Nod,Solution):- s(Nod,Nod1),\+ (member(Nod1,Drum)),
                                       depthfirst([Nod|Drum],Nod1,Solution).
% ----------------------------------------
%Cautare de tip iterative deepening
% ----------------------------------------
cale(Nod,Nod,[Nod]).
cale(PrimNod,UltimNod,[UltimNod|Drum]):- cale(PrimNod,PenultimNod,Drum),
                                       s(PenultimNod,UltimNod),
                                       \+(member(UltimNod,Drum)).

depth_first_iterative_deepening(Nod,Solutie):- cale(Nod,NodScop,Solutie),
                                       scop(NodScop),!.

% ----------------------------------------
%Predicatele specifice problemei mutarii blocurilor
% ----------------------------------------
s(Lista_stive,Lista_stive_rez):- member(X,Lista_stive),X=[Varf|_],
                                       det_poz_el(Lista_stive,N,X),
                                       sterg_la_n(Lista_stive,Lista_stive_inter,N),
                                       member(Y,Lista_stive),
                                       det_poz_el(Lista_stive,N1,Y),N1\==N,
                                       adaug_la_n(Varf,Lista_stive_inter,Lista_stive_rez,N1).
% configuratia initiala a stivelor.
initial([[d],[a,b],[c]]).

% configuratia-scop a stivelor, cea pe care o cauta fiecare dintre algoritmii de cautare folositi.
scop([[],[a,b,c,d],[]]).
```

```
% Mai jos se gasesc trei posibilitati de apel pentru problema mutarii blocurilor. Alegeti cate una, in
% functie de ce algoritm doriti sa folositi

% Inlocuiti calea de mai jos cu una catre directorul in care lucrati voi
pb_bf:-tell('C:\\bloc_mut_ies_bf.txt'), initial(S),rezolva_b(S,Solutie),afisare(Solutie),told.

pb_df:-tell('C:\\bloc_mut_ies_df.txt'), initial(S),rezolva(S,Solutie),afisare(Solutie),told.

pb_df_id:-tell('C:\\block_mut_ies_df_id_.txt'), initial(S), depth_first_iterative_deepening(S,Solutie),
            afisare(Solutie),told.
% ----------------------------------------
%% END mutare blocuri
% ----------------------------------------
```

1. afisare(Solutie).

Solutie va fi o lista de forma urmatoare:
Solutie = [ [[d],[a,b],[c]], [[],[d,a,b],[c]], [[],[a,b],[d,c]], ... ]. Solutie contine, deci, toate mutarile necesare pentru a ajunge din C1 in C2. Scopul acestui predicat este acela de a afisa cat mai frumos fiecare dintre mutarile efectuate.
Exemplu de afisare:

```
    a
d   b   c
=   =   =

    d
    a
    b   c
=   =   =

    a   d
    b   c
=   =   =
```

2. det_poz_el(Lista_stive,N,X)
Daca Lista_stive este [[d],[a,b],[c]], iar X este [a,b], atunci acest predicat ar trebui sa intoarca N = 2.

3. sterg_la_n(Lista_stive,Lista_stive_inter,N)
Daca Lista_stive este [[d],[a,b],[c]], iar N este 2, atunci acest predicat ar trebui sa intoarca Lista_stive_inter =[[d],[b],[c]]

4. adaug_la_n(Varf,Lista_stive_inter,Lista_stive_rez,N1)
Daca Lista_stive_inter =[[d],[b],[c]], N1 = 3, Varf = a, atunci acest predicat ar trebui sa intoarca Lista_stive_rez = [[d],[b],[a,c]].