Deklaratív Programozás gyakorlat
Prolog programozás: listakezelés, gráfok

Az alábbi feladatok megoldásában, ha másként nem mondjuk, használhat segédeljárásokat, de ezekhez mindig adjon meg fejkommentet. Megoldásában mindig felhasználhatja az előző feladatokhoz megírt eljárásokat.

1. Beszúrás rendezett listába

```
% insert_ord(+RL0, +Elem, ?RL): Az RL szigorúan monoton növő számlista
% úgy áll elő, hogy az RL0 szigorúan növő számlistába beszúrjuk az Elem
% számot, feltéve hogy Elem nem eleme az RL0 listának; egyébként RL = RL0.

| ?- insert_ord([1,3,5,8], 6, L).
L = [1,3,5,6,8] ? ; no
| ?- insert_ord([1,3,5,8], 3, L).
L = [1,3,5,8] ? ; no
```

Megoldása legyen jobbrekurzív, használjon feltételes szerkezetet!

Megvalósítási ötletek:
- A predikátum két klózból álljon: az első az üres lista, a második a nem-üres lista esetét fedje le.
- A második klózban egy feltételes szerkezet segítségével ágazzon el háromfelé aszerint hogy RL0 feje és Elem milyen viszonyban van: <, =, vagy >. A jobbrekurzió érdekében az RL kimenő argumentumot behelyettesítő hívást hozza a rekurzív hívás elé.

A 'graph' adatstruktúrát a következő Mercury-szerű típusdefiníciókkal definiáljuk:

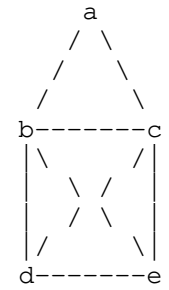
```
% :- type graph == list(edge).
% :- type edge ---> node-node.
% :- type node == atom.
```

Eszerint egy Prolog kifejezés a 'graph' típusba tartozik, ha X-Y alakú struktúrák listája, ahol X és Y névkonstansok (atomok).

Az [a1-b1,a2-b2,...,an-bn] 'graph' típusú kifejezés azt az irányítatlan gráfot írja le, amelynek csomópont-halmaza {a1,..,an,b1,...,bn}, és egy (irányítatlan) él van ai és bi között, minden i=1,...,n esetén. (Megjegyzés: az így megadott gráfoknak nyilván nem lehet izolált pontja.)

Például az [a-b,a-c], [a-c,b-a], [b-a,a-c], [c-a,a-b] stb. mind ugyanazt a (matematikai értelemben vett) irányítatlan gráfot írják le.

Az [a-b,a-c,b-c,b-d,b-e,c-d,c-e,d-e] kifejezés az alábbi gráfot írja le:



Gyerekkorukban találkozhattak azzal a feladattal, hogy ezt a gráfot egy folytonos vonallal rajzolják meg.

A későbbi példafutások érdekében vegye fel a programjába az alábbi tényállítást:

```
house([a-b,a-c,b-c,b-d,b-e,c-d,c-e,d-e]).
```

Egy 'graph' típusú [a1-b1,a2-b2,...,an-bn] Prolog listát folytonos vonalnak hívunk, ha b1=a2, b2=a3, ..., b(n-1) = an.

2a. Írjon egy graph/1 Prolog eljárást amely egy tetszőleges Prolog kifejezésről eldönti, hogy az a 'graph' típusba tartozik-e.

```
% graph(G): G egy 'graph' típusba tartozó Prolog kifejezés.

| ?- graph([]).
yes
| ?- graph([a-b,b-c]).
yes
| ?- graph([a-b,b-1]).
no
```

Megoldása legyen jobbrekurzív!

2b. Írjon egy same_edge/2 Prolog eljárást amely két 'edge' típusú Prolog kifejezésről eldönti, hogy azonos irányítatlan élet írnak-e le!

```
| ?- same_edge(a-b, a-b).
yes
| ?- same_edge(a-b, b-a).
yes
| ?- same_edge(a-b, b-b).
no
```

Az alábbi célok futtatásával vizsgálja meg, hogy az eljárás a (+,+) módtól különböző módokban is működik-e.

```
| ?- same_edge(a-b, E).

| ?- same_edge(E, a-b).

| ?- same_edge(E1, E2).
```

2c. Írjon egy same_graph0/2 Prolog jobbrekurzív eljárást amely két 'graph' típusú Prolog kifejezésről eldönti, hogy (matematikai értelemben) azonos gráfot írnak-e le!

```
% same_graph0(G1, G2): G1 és G2 azonos gráfot írnak le.

| ?- same_graph0([], []).
yes
| ?- same_graph0([a-b], [a-b]).
yes
| ?- same_graph0([a-b], [b-a]).
yes
| ?- same_graph0([a-b,b-c], [b-c,b-a]).
yes
| ?- same_graph0([a-b,a-b], [a-b]).
no
```

Megvalósítási ötletek:

- Használja a same_edge/2 segédeljárást.
- Használja a select/3 eljárást, amely elérhető a lists könyvtárban, illetve a 248. dián.

Vizsgálja meg, hogy helyesen és végesen működik-e az eljárás, ha csak az egyik argumentumot adjuk meg (és a másik változó). Például futtassa az alábbi két eljáráshívást:

```
| ?- same_graph0([a-b], G).

| ?- same_graph0(G, [a-b]).
```

Ezek közül az egyik az első megoldás után végtelen választási pontot hoz létre. A trace eljárás meghívásával kapcsolja be a nyomkövetést, futtassa a megfelelő eljáráshívást és értse meg, hogy miért jön létre végtelen választási pont. (Lásd még a 248. dia utolsó mondatát).

Becsülje meg az alábbi célsorozat megoldásainak számát:

```
| ?- house(G0), same_graph0(G, G0).
```

Az alábbi célsorozat futtatásával megállapíthatja a pontos megoldásszámot:

```
| ?- house(G0), findall(1, same_graph0(G, G0), Gs), length(Gs, N).
```

2d. Olvassa el a same_length/2 lists könyvtárbeli eljárás specifikációját, az SWI Prolog esetében pl. itt: https://www.swi-prolog.org/pldoc/doc_for?object=same_length/2 (A kék fejléc jobb szélén levő narancssárga körbe foglalt :- jelre kattintva megnézheti az eljárás -- végtelenül egyszerű -- Prolog kódját is.)

A same_length/2 segítségével írjon egy same_graph(G0, G) eljárást, amelynek jelentése ugyanaz, mint a same_graph0 eljárásé, de nem érzékeny az argumentumok sorrendjére: ha legalább az egyik argumentuma zárt végű lista, akkor véges a keresési tere.

Megvalósítási segítség: a same_graph egyetlen klózzal megvalósítható úgy, hogy a klóz törzse csak két hívásból áll.

2e. Írjon egy line/2 eljárást, amely eldönti, hogy egy adott gráf egy adott pontból kiinduló folytonos vonal-e.

```
% line(G, P):  A G gráf a P pontból kiinduló folytonos vonal.

| ?- line([], a).
yes
| ?- line([a-b], a).
yes
| ?- line([a-b,c-b], a).
no
| ?- line([a-b,b-c], a).
yes
| ?- line([a-b,b-c], c).
no
```

Vizsgálja meg, hogy a line eljárás hogy viselkedik, ha az első, G argumentum változó, illetve ha változókból álló (zárt végű) lista. Ehhez futtassa az alábbi célokat:

```
| ?- line(L, a).

| ?- length(L, 5), line(L, P).
```

Vegye észre, hogy ha L zárt végű, változókból álló lista (mint az utolsó példában), akkor a line(L, P) hívás létrehoz egy olyan folytonos vonalat, amelynek pontjai még változók (hála a Prolog "logikai" változó-fogalmának, azaz annak, hogy a változók az adatstruktúrák részei lehetnek). (*)

A jelen feladat hátralevő alpontjaiban az alábbi fejkommentnek megfelelő draw/2 Prolog eljárás különböző változatait készítjük el.

```
% draw(+G, -L): Az L folytonos vonal "megrajzolja" a G gráfot, azaz az
% L folytonos vonal ugyanazt a matematikai értelemben vett gráfot írja
% le, mint a G Prolog kifejezés.
```

Tehát a "| ?- draw(G, L)." hívás, ahol G adott és L egy változó, felsorolja L-ben az összes olyan folytonos vonalat, amely "megrajzolja" G-t.

```
| ?- draw([a-b,a-c], L).
L = [b-a,a-c] ? ;
L = [c-a,a-b] ? ;
no
| ?- draw([a-b,a-c,b-c,b-d,b-e,c-d,c-e,d-e], L), L = [d-e|_].
L = [d-e,e-b,b-a,a-c,c-b,b-d,d-c,c-e] ? ;
L = [d-e,e-b,b-a,a-c,c-d,d-b,b-c,c-e] ? ;
L = [d-e,e-b,b-c,c-a,a-b,b-d,d-c,c-e] ? ;
L = [d-e,e-b,b-c,c-d,d-b,b-a,a-c,c-e] ? ;
L = [d-e,e-b,b-d,d-c,c-a,a-b,b-c,c-e] ? ;
L = [d-e,e-b,b-d,d-c,c-b,b-a,a-c,c-e] ? ;
L = [d-e,e-c,c-a,a-b,b-c,c-d,d-b,b-e] ? ;
L = [d-e,e-c,c-a,a-b,b-d,d-c,c-b,b-e] ? ;
L = [d-e,e-c,c-b,b-a,a-c,c-d,d-b,b-e] ? ;
L = [d-e,e-c,c-b,b-d,d-c,c-a,a-b,b-e] ? ;
L = [d-e,e-c,c-d,d-b,b-a,a-c,c-b,b-e] ? ;
L = [d-e,e-c,c-d,d-b,b-c,c-a,a-b,b-e] ? ;
no
```

2f. Írja meg a draw0(G, L) eljárást "generál és ellenőriz" (generate and test) formában: generálja le a G gráffal matematikailag azonos L gráfokat, majd válassza ki ezek közül azokat, amelyek folytonos vonalat alkotnak!

Megvalósítási ötlet:

A same_graph0/2 eljárás segítségével sorolja fel az adott G-vel azonos L gráfokat (vigyázzon G és L megfelelő sorrendjére, vagy használja same_graph/2 eljárást), majd line/2 segítségével válassza ki azokat amelyek folytonos vonalak.

2g. Írja meg a draw1(G, L) eljárást "ellenőriz és generál" (test and generate) formában!

Megvalósítási ötlet: a 2e. variáns (*)-gal jelzett utolsó bekezdése szerint a line(L, _) eljárás véges lefutásához elegendő, hogy L hossza ismert legyen. Ennek biztosítására használja a same_length/2 eljárást, majd a draw0/2 törzsében levő két hívást fordított sorrendben hajtsa végre.

A draw1/2 eljárás kb 7500-szer gyorsabb mint a draw0/2!

2h. Írja meg a draw2/3 segédeljárást az alábbi fejkommentnek megfelelően!

```
% draw2(G, P, L): A G gráfot megrajzolja a P pontból kiinduló L
% folytonos vonal.
```

Megvalósítási ötlet: használja a select/3 és a same_edge/2 eljárásokat.

Az eljárás írásakor feltételezheti, hogy P ismert, de szinte biztos, hogy akkor is működik az eljárás, ha P nincs behelyettesítve.

Az utóbbi észrevételre építve írja meg a draw/2 eljárás újabb változatát draw2/2 néven, amely a draw2/3-ra vezeti vissza feladatot.

A draw2/2 eljárás lényegében azonos sebességű a draw1/2-vel.

2i. Írja meg a draw3(G, L) eljárást úgy, hogy csak a select/3 könyvtári eljárást használja, más segédeljárást nem.

3. (Otthoni feladat)

Írjon Prolog eljárást egy (irányítatlan) gráf fokszámlistájának előállítására. A fokszámlista típusa:

```
% :- type degrees == list(node_degree).
% :- type node_degree --> node - degree.
% :- type degree == int.
```

A fokszámlista tehát egy olyan lista, amelyenek elemei Cs-N alakú párok, ahol Cs a gráf egy csomópontja, és N a Cs csomópont fokszáma. A csomópontok tetszőleges sorrendben szerepelhetnek a fokszámlistában, de mindegyik pontosan egyszer.

```
% degree_list(G, Ds): A G gráf fokszámlistája Ds.

| ?- degree_list([b-a,a-c], Ds).
Ds = [b-1,a-2,c-1] ? ; no
```

4. (Otthoni feladat)

Írjon idraw/2 néven egy Prolog eljárást, amelynek jelentése azonos a 2. feladatban szereplő draw/2 eljárással! Törekedjék minél hatékonyabb megoldásra! Javaslat: használja a ugraphs könyvtárat.

Platónak hívunk egy olyan legalább kételemű listát, amely csupa azonos elemből áll. Az mondjuk, hogy MP egy L listában található maximális plató, ha
- MP az L folytonos része (azaz MP előáll úgy, hogy L elejéről és végéről 0 vagy több elemet elhagyunk),
- MP egy plató és
- MP maximális L-ben, azaz nem lehet sem balra sem jobbra a benne levőkkel azonos, közvetlenül szomszédos elemekkel kiterjeszteni.

Például az L = [a.b,a,c,c,c,b,b] listában két maximális plató van, a 4. pozición kezdődő [c,c,c] és a 7. pozición kezdődő [b,b] lista (a listaelemeket 1-től számozzuk).

Egy adott listában előforduló platók felsorolására használható az alábbi deklaratív, de nem hatékony (potenciálisan négyzetes költségű) Prolog eljárás:

```
:- use_module(library(lists), [append/2,last/2]).
:- use_module(library(aggregate), [forall/2]).

plato0(L, I, Len, X) :-     % Az L listában az I. pozición van egy
                            % Len hosszúságú, X elemekből képzett
                            % maximális plató                          HA
    L2 = [X,X|_],           % L2 két azonos X elemmel kezdődő lista    ÉS
    append([L1,L2,L3], L),  % L szétszedhető L1, L2, L3 részlistákra   ÉS
    forall( member(Y, L2),  % L2 minden Y eleme
            X = Y ),        %    azonos X-szel                         ÉS
    \+ L3 = [X|_],          % L3 nem X-szel kezdődő lista  (1)         ÉS
    \+ last(L1, X),         % L1 nem X-szel végződő lista  (2)         ÉS
    length(L2, Len),        % L2 hossza Len,                           ÉS
    length([a|L1], I).      % I = 1+(L1 hossza).

% Vegyük észre, hogy az (1) ill. (2) feltételek akkor is teljesülnek, ha
% az L3 ill. L1 listák üresek!
```

Az alábbi eljárások segítségével a fenti plato0/4 eljárással ekvivalens, de lényegesen hatékonyabb plato/4 predikátumot valósítunk majd meg.

---

5. Írjon Prolog eljárást amely a bemenetként megadott listáról eldönti, hogy egy platóval kezdődik-e, és ha igen, visszaadja a maximális plató hosszát és az ezutáni (maradék) elemek listáját.

```
% pl_kezdetu(+L, ?Len, ?M): Az atomokból álló L lista egy Len hosszúságú
% maximális platóval kezdődik, amelyet az M maradéklista követ.

| ?- pl_kezdetu([a,b,a,c,c,c,b,b], Len, M).
no
| ?- pl_kezdetu([c,c,c,b,b], Len, M).
Len = 3, M = [b,b] ? ; no
| ?- pl_kezdetu([b,b], Len, M).
Len = 2, M = [] ? ; no
| ?- pl_kezdetu([b], Len, M).
no
| ?- pl_kezdetu([], Len, M).
no
| ?-
```

6. Írjon olyan Prolog eljárást, amely felsorolja atomok egy adott listájában található maximális platókat, megadva a plató hosszát és az ismétlődő elemet.

```
% plato(L, Len, X): Az L listában található egy Len hosszúságú,
% X elemekből képzett maximális plató.

| ?- plato([a,b,b,b,b,a,a,c,b,b], Len, X).
Len = 4, X = b ? ;
Len = 2, X = a ? ;
Len = 2, X = b ? ;
no
```

7. (Otthoni feladat)

Az előző feladat kiterjesztéseként írjon olyan Prolog eljárást, amely felsorolja atomok egy adott listájában található maximális platókat, megadva a plató kezdőindexét (1-től számozva), hosszát és az ismétlődő elemet.

```
% plato(L, I, Len, X): Az L listában az I-edik elemtől kezdődően
% egy X elemekből képzett, Len hosszúságú maximális plató található.

| ?- plato([a,b,b,b,b,a,a,c,b,b], I, Len, X).
I = 2, Len = 4, X = b ? ;
I = 6, Len = 2, X = a ? ;
I = 9, Len = 2, X = b ? ;
no
```