Deklaratív programozás, 4. gyakorlat
Erlang programozás: struktúrák, visszalépés
2013. 10. 17.

I. RÉSZ: LISTAKEZELÉS

---

1. Számsorozat jobbrekurzívan (lists:seq/2)

```
%% @spec seq(F::integer(), T::integer()) -> S::[integer()].
%% S = [F, F+1, ..., T].

seq(10,13) =:= [10,11,12,13].
```

---

2. Lista kilapítása (lists:flatten/1)

```
%% @spec flatten(DeepList::list()) -> L::list().
%% A DeepList - tetszőleges mélységű beágyazott listákban tartalmazott -
%% elemeit az L listában "kibontva" tartalmazza úgy, hogy az L listában
%% már nem szerepel elemként további lista.

flatten([1,[2,3],[[[[4]]],[5,[6]]]]) =:= [1,2,3,4,5,6].

Megjegyzés: naív változathoz használható a lists:append/2 (++).
* Szorgalmi: append nélkül, flatten/2 segédfüggvény bevezetésével.
```

---

3. Mátrix részmátrixa

```
%% @spec kozepe(M::[[term()]]) -> M1::[[term()]].
%% M1 az M n*n-es négyzetes mátrix olyan (n/2)*(n/2) méretű részmátrixa,
%% mely az n/4+1. sor n/4+1. oszlopának elemétől kezdődik.

M=[[a,b,e,f],
   [c,d,g,h],
   [i,j,m,n],
   [k,l,o,p]].

kozepe(M) =:= [[d,g],[j,m]].

A megoldást valósítsa meg többféleképpen is:
    a) Használja fel listanézetben a lists:sublist/3 függvényt:
       %% @spec sublist(L1::list(), S::int(), L::int()) -> L2::list().
    b) Használja fel listanézetben az lists:nth/2 függvényt:
       %% @spec nth(N::int(), List::list()) -> Elem::term().
```

---

4. Mátrix középső elemei

```
%% @spec laposkozepe((M::[[term()]]) -> L::[term()].
%% L lista az M mátrix közepe elemeinek listája.

laposkozepe(M) =:= [d,g,j,m].

A megoldást valósítsa meg többféleképpen is:
    a) flatten/1 és kozepe/1 felhasználásával.
    b) Használja fel listanézetben az lists:nth/2 függvényt.
```

---

5. Részmátrix sor és oszlop elhagyásával

```
%% @spec pivot((M::[[term()]], R::int(), C::int()) -> M1::[[term()]].
%% M1 az M mátrix R. sorának és C. oszlopának elhagyásával keletkezik.

pivot(M,2,3) =:= [[a,b,f],[i,j,n],[k,l,p]].

A megoldáshoz használja fel listanézetben az lists:nth/2 függvényt.
```

---

6. (*) Szorgalmi: összetett számok

```
%% @spec osszetett(K::integer()) -> L::[integer()].
%% L tartalmazza az összetett számokat 4..K*K között,
%% ismétlődések megengedettek.

E módszer Eratoszthenész_szitája néven ismert: bejárjuk minden szám
többszöröseit, és megjelöljük őket összetettként. Felhasználható
lists:seq/3 függvény:
% seq(F, T, D) =:= [F, F+D, F+2D, ..., F+KD], ahol F+KD =< T < F+(K+1)D.

osszetett(5) =:= [4,6,8,10,12,14,16,18,20,22,24,
                  6,9,12,15,18,21,24,
                  8,12,16,20,24,
                  10,15,20,25].
```

---

7. (*) Szorgalmi: prímszámok

```
%% @spec primek(K::integer()) -> L::[integer()].
%% L tartalmazza a prímszámokat 2..K*K között.

primek(5) =:= [2,3,5,7,11,13,17,19,23].
```

---

8. Lista elemeinek egyezése

```
%% @spec all_different(Xs::[any()]) -> B::bool().
%% B igaz, ha az L lista minden eleme különbözik.

all_different([1,2,3,1]) =:= false.
all_different([1,2,3]) =:= true.

A megoldást valósítsa meg többféleképpen is:
  a) lists:member felhasználásával,
     ügyelve a rekurzív hívás lusta kiértékelésére;
  b) lists:usort felhasználásával.
```

---

9. Listák cipzárazása (lists:zip/2, lists:unzip/1)

```
%% @spec zip(Xs::[term()], Ys::[term()]) -> XYs::[{term(), term()}].
%% @spec unzip(XYs::[{term(), term()}]) -> {Xs::[term()], Ys::[term()]}.
%% XYs olyan párok listája, amelynek első eleme az Xs, második eleme
%% az Ys lista azonos pozíciójú eleme.

zip([1,2,3], [a,b,c]) =:= [{1,a},{2,b},{3,c}].
unzip([{1,a},{2,b},{3,c}]) =:= {[1,2,3], [a,b,c]}.
```

```
-------------------------------------------------------------------------------
                        II. RÉSZ: 8 VEZÉR A SAKKTÁBLÁN
-------------------------------------------------------------------------------


 Feladat: 8 vezér (királynő) elhelyezése a sakktáblán úgy, hogy ne üssék
 egymást. Az alábbi feladatok egy lehetséges megoldás felépítéséhez vezetnek.

 %% @type sor()    = 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8.
 %% @type oszlop() = 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8.  % abcdefgh helyett.
 %% @type jelolt() = [oszlop()].

 Mivel egy megoldásban bármely sorban vezér csak     . . . X . . . .
 a többitől különböző oszlopban lehet, ezért a      . . . . . . . X .
 megoldások  az  oszlopindexek  permutációival      . . X . . . . .
 leírhatók. Vagyis egy megoldás egy olyan lista,    . . . . . . . X
 amelynek ha az i. eleme j,  akkor a sakktábla      . X . . . . . .
 i. sorában a j. oszlopban van vezér.  Például      . . . . X . . .
 jobbra a [4,7,3,8,2,5,1,6]  megoldás látható.      X . . . . . . .
 Ezzel automatikusan kizárjuk,  hogy két vezér      . . . . . X . .
 azonos oszlopban vagy sorban legyen, csak azt      (1 2 3 4 5 6 7 8)
 kell vizsgálni, hogy átlósan ütésben vannak-e.     (a b c d e f g h)

-------------------------------------------------------------------------------
10. Számpárok összeg-különbség párjainak előállítása (koordináta-transzformáció)

    %% @spec atlok([{R::sor(), C::oszlop()}]) ->
    %%           [Atlok::{integer(), integer()}].
    %% Minden {R,C} párhoz egy Atlok számpárt rendel, amelynek első eleme
    %% R+C, második eleme R-C.
    %% Ezzel egy {sor(),oszlop()} koordinátához rendelhető, hogy "hányadik"
    %% átlóban van: az összegnél balról, a különbségnél jobbról számítva.
    %% Egy 3x3-as táblán az  R+C értékek:  R-C értékek:  Átlók:
    %%                       2 3 4         0 -1 -2       {2,0} {3,-1} {4,-2}
    %%                       3 4 5         1  0 -1       {3,1} {4,0}  {5,-1}
    %%                       4 5 6         2  1  0       {4,2} {5,1}  {6,0}

    atlok([{1,1}, {2,3}]) =:= [{2,0}, {5,-1}].

-------------------------------------------------------------------------------
11. Ütések ellenőrzése

    %% @spec nem_utik_egymast(Js::jelolt()) -> B::bool().
    %% B igaz, ha a vezérek Js-beli elhelyezése helyes, vagyis nem ütik
    %% egymást egy length(Js) sorból és lists:max(Js) oszlopból álló táblán.
    Felhasználandó: all_different/1, seq/2, zip/2, atlok/1, unzip/1.

    nem_utik_egymast([1,3,5]) =:= true.
    nem_utik_egymast([1,5,3]) =:= false.

-------------------------------------------------------------------------------
12. 1..8 számok összes permutációjának felsorolása

    %% @spec perms() -> Ps::[jelolt()].
    %% Ps az 1..8 számok összes lehetséges permutációját tartalmazó lista.

    perms() =:= [[1,2,3,4,5,6,7,8], [1,2,3,4,5,6,8,7] | ... ].

-------------------------------------------------------------------------------
13. 8 vezér -- kimerítő keresés (generate and test)

    Oldjuk meg a feladatot kimerítő kereséssel: először generáljuk az
    összes permutációt, majd kiszűrjük azokat, amelyek helyes megfejtések.
    Felhasználandó: listanézet, perms/0, nem_utik_egymast/1.

    %% @spec vezerek8_1() -> [jelolt()].

    vezerek8_1() =:= [ [1,5,8,6,3,7,2,4] | ... ].

    length(vezerek8_1()) =:= 92.

-------------------------------------------------------------------------------
14. 8 vezér -- hatékonyabb megoldás

    Permutációk generálása közben is végzünk szűrést a részmegoldásra:
    ha már K=<8 sorhoz rendeltünk oszlopot, a K méretű részmegoldást is
    ellenőrizhetjük a nem_utik_egymast/1 függvénnyel.
    Nem használható fel a perms/0 függvény, hiszen a generátorok közé kell
    a szűréseket betenni.

    _ = statistics(runtime),           % idő lekérdezése
    V1 = vezerek8_1(),                 % V1 az összes megoldás
    {_,Time1} = statistics(runtime),   % utolsó lekérdezés óta eltelt idő
    V2 = vezerek8_2(),                 % V2 az összes megoldás
    {_,Time2} = statistics(runtime),   % utolsó lekérdezés óta eltelt idő
    lists:sort(V1) =:= lists:sort(V2)  % sorrendet nem kötjük meg
        andalso Time2 < Time1.         % de bizonyosan gyorsabb

-------------------------------------------------------------------------------
15. A sakktábla mérete 8 helyett N (paraméter). Keressük a megoldásokat
    kimerítő kereséssel az 1..N számok permutációi között.

-------------------------------------------------------------------------------
16. N méretű sakktábla hatékonyabb megoldása: a részmegoldásokat is
    ellenőrizzük.

----- $LastChangedDate: 2013-10-17 10:16:28 +0200 (cs, 17 okt 2013) $ ------
```