

Nagyhatékonyságú Deklaratív Programozás
2. gyakorlat

2013. 10. 03.

Ha a feladat szövege másként nem rendelkezik, írd meg az alábbi fejkommenteknek megfelelő Prolog eljárásokat! Törekedj minél hatékonyabb megoldásra! Használd a SICStus CLPQ könyvtár eljárásait!

1. Szélsőérték-számolás

```
% dilemma(+Osszeg, -Max):  
% Osszeg forinttal a zsebünkben, legfeljebb Max db boldogságpontot tudunk  
% összegyűjteni a következő feltételek betartásával:  
% Egy sör ára 200 Ft es 8 boldogságpontot nyújt.  
% Egy zsíroskenyér ára 100 Ft es 1 boldogságpontot nyújt.  
% Viszont a sör csak akkor esik jól, ha legalább feleannyi  
% zsíroskenyeret is megeszünk hozzá. Zsíroskenyérből törtrészt is vehetünk,  
% de sört csak darabonként árulnak.
```

```
| ?- dilemma(10, Max).  
Max = 1/10 ? ;  
no  
% source_info  
| ?- dilemma(210, Max).  
Max = 21/10 ? ;  
no  
% source_info  
| ?- dilemma(260, Max).  
Max = 43/5 ? ;  
no  
% source_info  
| ?- dilemma(800, Max).  
Max = 26 ? ;  
no  
% source_info  
| ?-
```

2. Egyszerűsített Sudoku

```
% sudoku_simple(?Matrix, +N):  
% Matrix egy N*N-es mátrix amely pozitív, N-nél nem nagyobb számokból áll.  
% Minden sorban és oszlopban a számok páronként különbözőek.  
% Minden sor és oszlop összege  $N*(N+1)/2$ .  
% Az eljárás ne hozzon létre választási pontot!
```

```
| ?- sudoku_simple(M, 2).  
M = [[_A,_B],[_C,_D]],  
{_D=3-_C}, {_B=_C}, {_A=3-_C}, {_C=<2}, {_C=\=3/2}, {_C>=1} ? ;  
no
```

Az összes a fenti feltételnek megfelelő mátrix felsorolásához írd meg az alábbi segédeljárást!

```
% place_sudoku(+Matrix, +N):  
% Matrix minden eleme 1 és N közé eső egész szám (CLPQ ábrázolásban!).
```

A zárójeles megjegyzésre azért van szükség, hogy az alábbi formában soroltathassuk fel a sudoku_simple/2 összes megoldását:

```
| ?- sudoku_simple(M, 2), place_sudoku(M, 2).  
M = [[1,2],[2,1]] ? ;  
M = [[2,1],[1,2]] ? ;  
no
```

3. Fibonacci

```
3a. % fibonacci0(?L, +N, +Max):
    % L pozitív számok N hosszú listája, N >= 3.
    % Az L lista elemei nem lehetnek nagyobbak Max-nál.
    % A lista bármely három A,B,C egymást követő elemére teljesül,
    % hogy C = A + B.
    % Az eljárás ne hozzon létre választási pontot!

    | ?- fibonacci0(L, 3, 2).
L = [_A,_B,_C], {_C=_A+_B}, {_A+_B=<2}, {_A>0}, {_B>0} ? ;
no

3b. % fibonacci(?L, +N, +Max):
    % Teljesülnek a fenti fibonacci0/3 által előírt feltételek, továbbá az
    % a feltétel is, hogy L elemei egész számok.

    Az eljárás sorolja fel az összes megoldást, a lehető legkevesebb
    választási pont létrehozásával!

    | ?- fibonacci(L, 10, 100).
L = [1,1,2,3,5,8,13,21,34,55] ? ;
L = [1,2,3,5,8,13,21,34,55,89] ? ;
L = [2,1,3,4,7,11,18,29,47,76] ? ;
L = [3,1,4,5,9,14,23,37,60,97] ? ;
no

    Tipp: Írd meg a between/3 eljárás CLPQ változatát, hogy ne kelljen a
    CLPQ racionálisok és Prolog egészek között konvertálni.

3c. Szorgalmi feladat: valósítsd meg az alábbi predikátumot, úgy, hogy
    annak a törzsében a fenti fibonacci0/3 mellett egyetlen CLPQ könyvtári
    eljáráshívás szerepel!

    % fibonacci(N, F): F a klasszikus Fibonacci sorozat (1 1 2 3 5 ...)
    % N. tagja, 1-től számozva, N >= 3.

    | ?- fibonacci(11, F).
F = 89 ? ;
no
    | ?- fibonacci(100, F).
F = 354224848179261915075 ? ;
no
```