Deklaratív Programozás 1. gyakorlat
Prolog programozás: struktúrák, listák

2010.09.13 és 2010.09.15.

Az alábbi példasorban a (bináris) fa adatstruktúra alatt egy olyan Prolog kifejezést értünk, amely lehet

- levél, azaz egy ‘leaf’ nevű struktúra, amelynek egyetlen argumentuma egy egész szám, pl. leaf(1), leaf(2); vagy
- csomópont, azaz egy node nevű kétargumentumú struktúra, amelynek mindkét argumentuma ‘fa’, pl. node(leaf(1),leaf(2)).

Az ilyen Prolog kifejezések tömör jellemzésére a következő kommentet használhatjuk:

```
% :- type fa ---> leaf(int) ; node(fa,fa).
```

Famintának hívunk egy Prolog kifejezést, ha egy a fenti módon definiált fakifejezésbol származtatható úgy, hogy a levelekben az egész számok helyére különböző változókat írunk.

Az egészlista olyan lista, melynek elemei egészek. Ez a következő kommenttel adható meg:

```
% :- type egészlista ---> [] ; [int|egészlista].
```

Listamintának hívunk egy olyan (zárt) listakifejezést, amelyben az elemek különböző változók.

Írjon olyan Prolog eljárást, amely megfelel az adott fejkommentnek. A feladat megoldásához felhasználhat korábbi sorszámú feladatokban definiált eljárásokat. Ha ilyeneken kívül is szükséges segédeljárás definiálása, azt külön jelezzük.

A fejkommentekben használt ún. módjelölések magyarázata:

*: az argumentum tömör bemenő, azaz nem lehet változó, és nem is tartalmazhat változót;

+: az argumentum bemenő, azaz nem lehet változó, de tartalmazhat változót (természetesen csak akkor, ha az argumentum egy struktúra);

−: az argumentum kimenő, azaz változó;

?: az argumentum lehet kimenő és bemenő is.

---

1. Fa csomópontjainak megszámolása

Egy fa csomópontjainak száma a benne előforduló node/2 struktúrák száma.

```
% fa_pontszama(*Fa, −N): A Fa bináris fa csomópontjainak száma N.

| ?− fa_pontszama(node(leaf(1),node(leaf(2),leaf(3))), N).
N = 2 ? ; no
| ?− fa_pontszama(node(leaf(1),node(leaf(2),node(leaf(4),leaf(3)))), N).
N = 3 ? ; no
```

Gondolja meg, hogy megoldása működőképes−e a fa_pontszama(*Fa, ?N) módban, azaz akkor is, ha a második argumentum nem változó, hanem adott egész:

```
| ?− fa_pontszama(node(leaf(1),node(leaf(2),node(leaf(4),leaf(3)))), 3).
yes
| ?− fa_pontszama(node(leaf(1),node(leaf(2),node(leaf(4),leaf(3)))), 2).
no
```

2. Fa mélységének meghatározása

Egy fa mélységén az egymásba skatulyázott node struktúrák maximális számát értjük.

```
% fa_melysege(*Fa, ?M): A Fa bináris fa mélysége M.

| ?− fa_melysege(node(node(leaf(1),leaf(4)),node(leaf(2),leaf(3))), N).
N = 2 ? ; no
| ?− fa_melysege(node(leaf(1),node(leaf(2),node(leaf(4),leaf(3)))), N).
N = 3 ? ; no
```

Tipp: az aritmetikai kifejezésekben megengedett a max függvény használata, pl | ?− X is max(2,3)+1. −−−> X = 4 ? ; no

3. Lista hosszának meghatározása

Egy lista hosszának az elemei számát nevezzük.

```
% lista_hossza(*Lista, −Hossz): A Lista egészlista hossza Hossz.

| ?− lista_hossza([1,3,5], H).
H = 3 ? ; no
```

4. Fa minden levélértékének növelése

```
% fa_noveltje(*Fa0, ?Fa): Fa úgy áll elő a Fa0 bináris fából, hogy az
% utóbbi minden egyes levelében levő értéket 1−gyel megnöveljük.

| ?− fa_noveltje(node(leaf(1),node(leaf(2),leaf(3))), Fa).
Fa = node(leaf(2),node(leaf(3),leaf(4))) ? ; no
```

5. Egészlista minden elemének növelése

```
% lista_noveltje(*L0, ?L): Az L egészlista úgy áll elő az L0
% egészlistából, hogy az utóbbi minden egyes elemét 1−gyel megnöveljük.

| ?− lista_noveltje([1,5,2], L).
L = [2,6,3] ? ; no
```

6. Egy fa leveleiben található értékek felsorolása

```
% fa_levelerteke(*Fa, −Ertek): A Fa bináris fa egy levelében található
% érték az Ertek.
```

Az eljárás nemdeterminisztikus módon sorolja fel az összes levélértéket. A felsorolás sorrendjére nem teszünk megkötést.

```
| ?− fa_levelerteke(node(leaf(1),node(leaf(2),leaf(3))), E).
E = 1 ? ; E = 2 ? ; E = 3 ? ; no
```

Gondolja meg, hogy a predikátum klózai sorrendjének változtatásakor hogyan változik a felsorolás sorrendje!

7. Egy fa részfáinak a felsorolása

Egy fa (nem feltétlenül valódi) részfájának nevezzük saját magát, valamint − ha a fa egy csomópont − akkor a bal és jobboldali ág részfáit.

```
% fa_reszfaja(*Fa, −Resz): Resz a Fa bináris fa részfája.
```

A fenti eljárás nemdeterminisztikus, azaz többféleképpen sikerül: a Resz változóban fel kell sorolnia a Fa összes részfáját. A felsorolás sorrendjére nem teszünk megkötést.

```
| ?− fa_reszfaja(node(leaf(1),node(leaf(2),leaf(3))), Fa).
Fa = node(leaf(1),node(leaf(2),leaf(3))) ? ;
Fa = leaf(1) ? ;
Fa = node(leaf(2),leaf(3)) ? ;
Fa = leaf(2) ? ;
Fa = leaf(3) ? ; no
```

Gondolja meg, hogy a predikátum klózai sorrendjének változtatásakor hogyan változik a felsorolás sorrendje!

A fa_reszfaja eljárás felhasználásával írja meg a 6. feladat megoldását, fa_levelerteke2 néven!

8. Egy lista prefixumainak a felsorolása

Egy L n-elemű lista prefixumának nevezzünk egy listát, ha az az L első k elemét tartalmazza (az L-beli sorrend megtartásával), ahol 0 =< k =< n.

```
% lista_prefixuma(*L0, -L): L az L0 egészlista prefixuma.
```

A fenti eljárás nemdeterminisztikus, azaz többféleképpen sikerül: az L változóban fel kell sorolnia a L0 összes prefixumát. A felsorolás sorrendjére nem teszünk megkötést.

```
| ?- lista_prefixuma([1,4,2], Sz).
Sz = [1,4,2] ? ;
Sz = [1,4] ? ;
Sz = [1] ? ;
Sz = [] ? ; no
```

Gondolja meg, hogy a predikátum klózai sorrendjének változtatásakor hogyan változik a felsorolás sorrendje!

9. Egy lista utolsó elemének meghatározása

```
% lista_utolso_eleme(*L, ?Ertek): A Lista egészlista utolsó eleme Ertek.

| ?- lista_utolso_eleme([5,1,2,8,7], E).
E = 7 ? ; no
```

10. Egy fa legbaloldalibb levélértékének meghatározása

```
% fa_balerteke(*Fa, ?Ertek): A Fa bináris fa legbaloldalibb levelében
% az Ertek egész érték szerepel.

| ?- fa_balerteke(node(node(leaf(1),leaf(4)),node(leaf(2),leaf(3))), E).
E = 1 ? ; no
```

11. Egy fa legjobboldalibb levélértékének meghatározása

```
% fa_jobberteke(*Fa, ?Ertek): A Fa bináris fa legjobboldalibb levelében
% az Ertek egész érték szerepel.

| ?- fa_jobberteke(node(node(leaf(1),leaf(4)),node(leaf(2),leaf(3))), E).
E = 3 ? ; no
```

12. Egy fa rendezettségének eldöntése

Egy fát rendezettnek mondunk, ha a levelek értékei, balról jobbra haladva szigorúan monoton növő sorozatot alkotnak.

```
% fa_rendezett(*Fa): A Fa bináris fa rendezett.

| ?- fa_rendezett(node(node(leaf(1),leaf(4)),node(leaf(2),leaf(3)))).
no
| ?- fa_rendezett(node(node(leaf(1),leaf(3)),
                       node(leaf(5),node(leaf(6),leaf(9))))).
yes
```

A megoldásban célszerű az előző két feladat eljárásait segédeljárásként felhasználni, így nem szükséges további segédeljárást definiálni.

Keressen hatékonyabb megoldást is, amelyben a fastruktúrát csak egyszer járja be! Ebben feltételezheti, hogy a fa csak pozitív számokat tartalmaz (fa_rendezett2). Ehhez a megoldáshoz szükség lehet egy megfelelő segédeljárásra.

13. Fa tükörképének képzése

```
% fa_tukorkepe(*Fa0, ?Fa): Fa a Fa0 bináris fa tükörképe.

| ?- fa_tukorkepe(node(node(leaf(1),leaf(4)),node(leaf(2),leaf(3))), Fa).
Fa = node(node(leaf(3),leaf(2)),node(leaf(4),leaf(1))) ? ; no
| ?- fa_tukorkepe(node(node(leaf(1),leaf(4)),node(leaf(4),leaf(1))), Fa).
Fa = node(node(leaf(1),leaf(4)),node(leaf(4),leaf(1))) ? ; no
```

14. Fa tükörszimmetrikus voltának ellenőrzése

```
% fa_tukros(*Fa): A Fa bináris fa tükörszimmetrikus.

| ?- fa_tukros(node(node(leaf(1),leaf(4)),node(leaf(4),leaf(1)))).
yes
| ?- fa_tukros(node(node(leaf(1),leaf(4)),node(leaf(2),leaf(3)))).
no
```

15. Adott elemszámú listaminta előállítása

```
% listaminta_adott_hosszal(-Listaminta, +Hossz): A Listaminta hossza Hossz.

| ?- listaminta_adott_hosszal(L, 4).
L = [_A,_B,_C,_D] ? ; no
```

16. Adott számnál kisebb-egyenlő mélységű faminták felsorolása

```
% faminta_melysege_max(-Faminta, +MaxMelyseg):
% A Faminta binárisfa-minta mélysége =< MaxMelyseg.
```

Az eljárás nemdeterminisztikus módon sorolja fel az összes az adott mélységnél kisebb-egyenlő mélységű famintákat. A felsorolás sorrendjére nem teszünk megkötést.

```
| ?- faminta_melysege_max(FM, 2).
FM = leaf(_A) ? ;
FM = node(leaf(_A),leaf(_B)) ? ;
FM = node(leaf(_A),node(leaf(_B),leaf(_C))) ? ;
FM = node(node(leaf(_A),leaf(_B)),leaf(_C)) ? ;
FM = node(node(leaf(_A),leaf(_B)),node(leaf(_C),leaf(_D))) ? ; no
```

17. Adott számú csomópontból álló faminták felsorolása

```
% faminta_adott_pontszammal(-Faminta, +Pontszam): A Faminta
% binárisfa-minta csomópontjainak száma Pontszam.
```

Az eljárás nemdeterminisztikus módon sorolja fel az összes az adott számú csomópontot tartalmazó famintákat. A felsorolás sorrendjére nem teszünk megkötést. Segédeljárásra szükség lehet.

```
| ?- faminta_adott_pontszammal(FM, 3).
FM = node(leaf(_A),node(leaf(_B),node(leaf(_C),leaf(_D)))) ? ;
FM = node(leaf(_A),node(node(leaf(_B),leaf(_C)),leaf(_D))) ? ;
FM = node(node(leaf(_A),leaf(_B)),node(leaf(_C),leaf(_D))) ? ;
FM = node(node(leaf(_A),node(leaf(_B),leaf(_C))),leaf(_D)) ? ;
FM = node(node(node(leaf(_A),leaf(_B)),leaf(_C)),leaf(_D)) ? ;
no
```