

```

1 /* Deklaratív programozás, 2. gyakorlat
2 Prolog programozás
3
4 MEGOLDÓKULCS
5
6 -----
7
8 "A" Témakör: a Prolog alapelemei
9 =====
10
11 A1.
12
13 [a*b-X,c+X]=[Y-3|Z]
14
15 bal: .(-(*(a,b),X),.(+(c,X),[]))
16 jobb: .(- (Y, 3),Z )
17
18 ----> X=3, Y=a*b, Z=[c+3]
19
20 A2.
21
22 g(V*W,[1*2+3|Z])=g(K,[K+L,L])
23
24 bal: g(*(V,W),.(+(*(1,2),3),Z ))
25 jobb: g(K, .(+ (K, L),.(L,[])))
26
27 ----> K=1*2, L=3, V=1, W=2, Z=[3]
28
29 A3.
30
31 s([a]-X/2,[X|Z])=s(Z-Y,[f,C])
32
33 bal: s(-.(a,[]),/(X,2)),.(X,Z ))
34 jobb: s(- (Z, Y), .(f,.(C,[])))
35
36 ----> C=a, X=f, Y=f/2, Z=[a]
37
38 A4.
39
40 [A+B,C*2|T]=[x/C+2,y*B,B-C]
41
42 bal: .(+ (A, B),.(*(C,2),T ))
43 jobb: .(+ (/ (x,C),2),.(*(y,B),.(-(B,C),[])))
44
45 ----> A=x/y, B=2, C=y, T=[2-y]
46
47 A5., A7., A8. lásd külön lapon.
48
49
50
51

```

```

52 "P". Témakör: programok írása
53 =====
54 */
55
56 % 1. Számsorozat generálása
57
58 % % seq(+N, +M, -L): Az L lista M-N+1 hosszú, elemei 1 különbségű
59 % % számtani sorozatot alkotnak, és L első eleme (ha van) N,
60 % % ahol N és M egész számok.
61
62 % | ?- seq(2, 4, L).
63 % L = [2,3,4] ? ; no
64 % | ?- seq(4, 2, L).
65 % no
66 % | ?- seq(4, 3, L).
67 % L = [] ? ; no
68 % | ?- seq(-4, -2, L).
69 % L = [-4,-3,-2] ? ; no
70
71 % seq(+N, +M, -L): Az L lista M-N+1 hosszú, elemei 1 különbségű számtani
72 % sorozatot alkotnak, és L első eleme (ha van) N, ahol N és M egész számok.
73 seq(N, M, []) :-
74     M =: N - 1.
75 seq(N, M, [N|Seq]) :-
76     M >= N,
77     N1 is N+1,
78     seq(N1, M, Seq).
79
80
81 % 2. Számintervallum felsorolása
82
83 % % max(+N, ?X): X egy egész szám, melyre 0 < X =< N, ahol N adott
84 % % pozitív egész szám. Az eljárás a fenti feltételeknek megfelelő X
85 % % számokat sorolja fel. A felsorolás sorrendjére nem teszünk megkötést.
86
87 % | ?- max(1,X).
88 % X = 1 ? ; no
89 % | ?- max(4,X).
90 % X = 4 ? ; X = 3 ? ; X = 2 ? ; X = 1 ? ; no
91 % | ?- max(4,3).
92 % yes
93 % | ?- max(4,5).
94 % no
95
96 % max(+N, ?X): X egy egész szám, melyre 0 < X =< N.
97 max(N, N) :-
98     N > 0.
99 max(N, X) :-
100     N > 1,
101     N1 is N-1,
102     max(N1, X).
103
104

```

```

105 % 3. Hatványozás
106
107 % % hatv(+A, +E, -H): H = A ^ E, ahol A egész szám, E >= 0 egész szám.
108
109 % | ?- hatv(3, 5, X).
110 % X = 243 ? ; no
111
112 % hatv(+A, +E, -H): H = A ^ E, ahol A egész szám, E >= 0 egész szám.
113 hatv(A, N, H) :-
114     N > 0,
115     N1 is N-1,
116     hatv(A, N1, H1),
117     H is A*H1.
118 hatv(_A, 0, 1).
119
120
121 % 4. Fa csomópontjainak megszámlálása
122
123 % Egy fa csomópontjainak száma a benne előforduló node/2 struktúrák
124 % száma.
125
126 % % fa_pontszama(*Fa, -N): A Fa bináris fa csomópontjainak száma N.
127
128 % | ?- fa_pontszama(node(leaf(1),node(leaf(2),leaf(3))), N).
129 % N = 2 ? ; no
130 % | ?- fa_pontszama(node(leaf(1),node(leaf(2),node(leaf(4),leaf(3))),
131 % N).
132 % N = 3 ? ; no
133
134 % fa_pontszama(+Fa, -N): A Fa bináris fa csomópontjainak száma N.
135 fa_pontszama(leaf(_), 0).
136 fa_pontszama(node(L,R), P) :-
137     fa_pontszama(L, LP),
138     fa_pontszama(R, RP),
139     P is LP+RP+1.
140
141
142 % 5. Fa minden levélértékének növelése
143
144 % % fa_noveltje(*Fa0, ?Fa): Fa úgy áll elő a Fa0 bináris fából, hogy az
145 % utóbbi minden egyes levelében levő értéket 1-gyel megnöveljük.
146
147 % | ?- fa_noveltje(node(leaf(1),node(leaf(2),leaf(3))), Fa).
148 % Fa = node(leaf(2),node(leaf(3),leaf(4))) ? ; no
149
150 % fa_noveltje(+Fa0, ?Fa): Fa úgy áll elő a Fa0 bináris fából, hogy az
151 % utóbbi minden egyes levelében levő értéket 1-gyel megnöveljük.
152 fa_noveltje(leaf(X), leaf(Y)) :-
153     Y is X+1.
154 fa_noveltje(node(L,R), node(NL,NR)) :-
155     fa_noveltje(L, NL),
156     fa_noveltje(R, NR).
157
158

```

```

159 % 6. Lista hosszának meghatározása
160
161 % Egy lista hosszának az elemei számát nevezzük.
162
163 % % lista_hossza(*Lista, -Hossz): A Lista egészlista hossza Hossz.
164
165 % | ?- lista_hossza([1,3,5], H).
166 % H = 3 ? ; no
167
168 % lista_hossza(+Lista, -Hossz): A Lista egészlista hossza Hossz.
169 lista_hossza([], 0).
170 lista_hossza(_[L], H) :-
171     lista_hossza(L, H0),
172     H is H0+1.
173
174 % 6*. (szorgalmi, otthoni feladat) Lista hosszának meghatározása --
175 % jobbrekurzív változat
176
177 % % lista_hossza2(*Lista, -Hossz): A Lista egészlista hossza Hossz.
178 % % Jobbrekurzív változat
179 % Segédeljárás szükséges.
180
181 % lista_hossza2(+Lista, +H0, H): A Lista egészlista hossza H-H0.
182 lista_hossza2([], H0, H0).
183 lista_hossza2(_[L], H0, H) :-
184     H1 is H0+1,
185     lista_hossza2(L, H1, H).
186
187 % lista_hossza2(+Lista, -Hossz): A Lista egészlista hossza Hossz.
188 lista_hossza2(Lista, Hossz) :-
189     lista_hossza2(Lista, 0, Hossz).
190
191
192 % 7. Egészlista minden elemének növelése
193
194 % % lista_noveltje(*L0, ?L): Az L egészlista úgy áll elő az L0
195 % egészlistából, hogy az utóbbi minden egyes elemét 1-gyel megnöveljük.
196
197 % | ?- lista_noveltje([1,5,2], L).
198 % L = [2,6,3] ? ; no
199
200 % lista_noveltje(+L0, ?L): Az L számlista úgy áll elő az L0
201 % számlistából, hogy az utóbbi minden egyes elemét 1-gyel megnöveljük.
202 lista_noveltje([], []).
203 lista_noveltje([X|L], [NX|NL]) :-
204     NX is X+1,
205     lista_noveltje(L, NL).
206
207
208 % 8. Egy lista utolsó elemének meghatározása
209
210 % % lista_utolso_eleme(*L, ?Ertek): Az L egészlista utolsó eleme Ertek.
211
212 % | ?- lista_utolso_eleme([5,1,2,8,7], E).
213 % E = 7 ? ; no
214
215 % lista_utolso_eleme(+L, ?Ertek): Az L egészlista utolsó eleme Ertek.
216 lista_utolso_eleme([E], E).
217 lista_utolso_eleme(_[L], E) :-
218     lista_utolso_eleme(L, E).
219
220

```

```

221 % 9. Egy fa leveleiben található értékek felsorolása
222
223 % % fa_levelerteke(*Fa, -Ertek): A Fa bináris fa egy levelében található
224 % % érték az Ertek.
225
226 % Az eljárás nondeterminisztikus módon sorolja fel az összes
227 % levélértéket. A felsorolás sorrendjére nem teszünk megkötést.
228
229 % | ?- fa_levelerteke(node(leaf(1),node(leaf(2),leaf(3))), E).
230 % E = 1 ? ; E = 2 ? ; E = 3 ? ; no
231
232 % fa_levelerteke(*Fa, -Ertek): A Fa bináris fa egy levelében található
233 % érték az Ertek.
234 fa_levelerteke(leaf(E), E).
235 fa_levelerteke(node(L,_), E) :-
236     fa_levelerteke(L, E).
237 fa_levelerteke(node(_,R), E) :-
238     fa_levelerteke(R, E).
239
240
241 % 10. Egy fa részfáinak a felsorolása
242
243 % Egy fa (nem feltétlenül valódi) részfájának nevezzük saját magát,
244 % valamint - ha a fa egy csomópont - akkor a bal és jobboldali ág
245 % részfáit.
246
247 % % fa_reszfaja(*Fa, -Resz): Resz a Fa bináris fa részfája.
248
249 % A fenti eljárás nondeterminisztikus, azaz többféleképpen sikerül:
250 % a Resz változóban fel kell sorolnia a Fa összes részfáját. A felsorolás
251 % sorrendjére nem teszünk megkötést.
252
253 % | ?- fa_reszfaja(node(leaf(1),node(leaf(2),leaf(3))), Fa).
254 % Fa = node(leaf(1),node(leaf(2),leaf(3))) ? ;
255 % Fa = leaf(1) ? ;
256 % Fa = node(leaf(2),leaf(3)) ? ;
257 % Fa = leaf(2) ? ;
258 % Fa = leaf(3) ? ; no
259
260 % Gondolja meg, hogy a predikátum klózái sorrendjének változtatásakor
261 % hogyan változik a felsorolás sorrendje!
262
263 % fa_reszfaja(+Fa, -Resz): Resz a Fa bináris fa részfája.
264 fa_reszfaja(Fa, Fa).
265 fa_reszfaja(node(L,_), Fa) :-
266     fa_reszfaja(L, Fa).
267 fa_reszfaja(node(_,R), Fa) :-
268     fa_reszfaja(R, Fa).
269
270 % A fa_reszfaja eljárás felhasználásával írja meg a 9. feladat
271 % megoldását, fa_levelerteke2 néven!
272
273 % fa_levelerteke2(+Fa, -Ertek): A Fa bináris fa egy levelében található
274 % érték az Ertek.
275 fa_levelerteke2(Fa, E) :-
276     fa_reszfaja(Fa, leaf(E)).
277

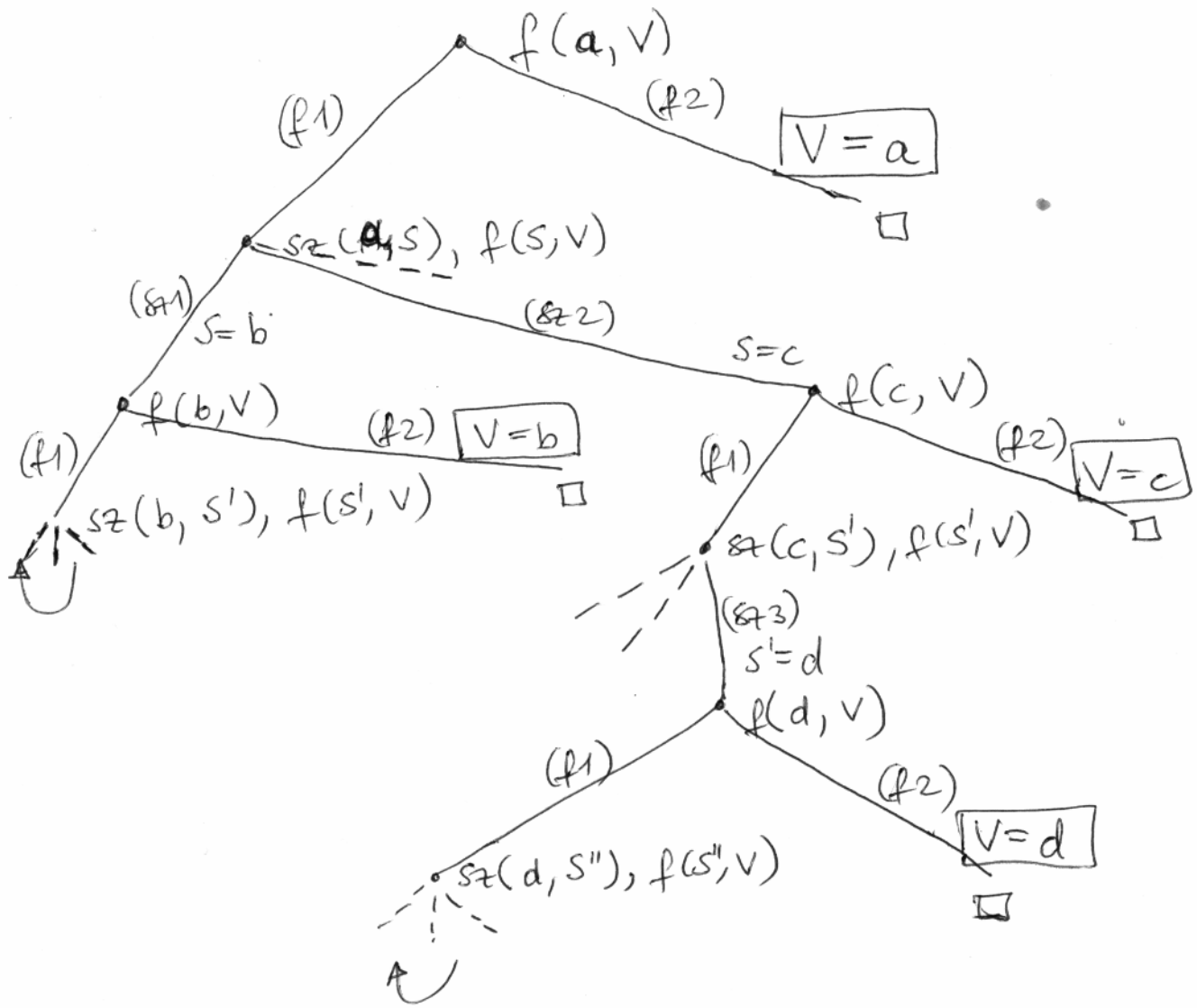
```

```

278 % 11. Egy lista prefixumainak a felsorolása
279
280 % Egy L n-elemű lista prefixumának nevezzünk egy listát, ha az az L
281 % első k elemét tartalmazza (az L-beli sorrend megtartásával),
282 % ahol 0 =< k =< n.
283
284 % % lista_prefixuma(*L0, -L): L az L0 egészlista prefixuma.
285
286 % A fenti eljárás nondeterminisztikus, azaz többféleképpen sikerül: az L
287 % változóban fel kell sorolnia a L0 összes prefixumát. A felsorolás
288 % sorrendjére nem teszünk megkötést.
289
290 % | ?- lista_prefixuma([1,4,2], Sz).
291 % Sz = [1,4,2] ? ;
292 % Sz = [1,4] ? ;
293 % Sz = [1] ? ;
294 % Sz = [] ? ; no
295
296 % Gondolja meg, hogy a predikátum klózái sorrendjének változtatásakor
297 % hogyan változik a felsorolás sorrendje!
298
299 % lista_prefixuma(*L0, -L): L az L0 egészlista prefixuma.
300 lista_prefixuma([X|L], [X|P]) :-
301     lista_prefixuma(L, P).
302 lista_prefixuma(_, []).

```

A5.



Teljesít a megoldás $V=b, U=d, V=c, V=a$

A7. A megoldás $V=a, U=b, V=c, V=d$

A8. Végtelen ciklus