

```

1 -module(dp12a_gy4).
2 -compile(export_all).
3 -author('patai@iit.bme.hu, hanak@iit.bme.hu, kapolnai@iit.bme.hu').
4 -vsn('$LastChangedDate: 2012-10-29 21:39:15 +0100 (Mon, 29 Oct 2012) $$').
5
6 %-----
7 %                               I. RÉSZ: FÁK
8 %-----
9
10 % 1.
11 fa_noveltje(level) ->
12     level;
13 fa_noveltje({C,Bfa,Jfa}) ->
14     {C+1, fa_noveltje(Bfa), fa_noveltje(Jfa)}.
15
16 % 2.
17 fa_tukorkepe(level) ->
18     level;
19 fa_tukorkepe({C,Bfa,Jfa}) ->
20     {C, fa_tukorkepe(Jfa), fa_tukorkepe(Bfa)}.
21
22 % 3. a)
23 fa_balerteke(level) ->
24     error;
25 fa_balerteke({C,level,_}) ->
26     {ok, C};
27 fa_balerteke({_,Bfa,_}) ->
28     fa_balerteke(Bfa).
29
30 % 3. b)
31 fa_jobberteke(level) ->
32     error;
33 fa_jobberteke({C,_,level}) ->
34     {ok, C};
35 fa_jobberteke({_,_,Jfa}) ->
36     fa_jobberteke(Jfa).
37
38 % 4.
39 rendezett_fa(level) ->
40     true;
41 rendezett_fa({C,Bfa,Jfa}) ->
42     case fa_jobberteke(Bfa) of
43     error -> true;
44     {ok,J} -> J < C
45     end andalso
46     rendezett_fa(Bfa) andalso
47     case fa_balerteke(Jfa) of
48     error -> true;
49     {ok,B} -> C < B
50     end andalso
51     rendezett_fa(Jfa).
52
53 % 4. kicsit hatékonyabb változatokban a fájl végén, kiegészítő anyag
54
55 % 5.
56 tartalmaz(_, level) ->
57     false;
58 tartalmaz(C, {C,_,_}) ->
59     true;
60 tartalmaz(C, {_,Bfa,Jfa}) ->
61     tartalmaz(C, Bfa) orelse tartalmaz(C, Jfa).
62
63 % 6.
64 elofordul(_, level) ->
65     0;
66 elofordul(C, {R,Bfa,Jfa}) ->
67     if C == R -> 1;
68     true -> 0
69     end
70     + elofordul(C, Bfa) + elofordul(C, Jfa).

```

```

71
72 % 7.
73 utak(Fa) -> utak(Fa, []).
74
75 % @spec utak(F::fa(), Eddigi::ut()) -> CimkezettUtak::[{C::term(), U::ut()}].
76 % A CimkezettUtak lista az F fa minden csomópontjához egy kételemű ennest
77 % társít, amelynek első eleme (C) a csp. címkéje, második eleme (U) az
78 % Eddigi útvonal és a csp. útvonala összefűzve.
79 utak(level, _) ->
80     [];
81 utak({C,Bfa,Jfa}, Eddigi) ->
82     Eddigil = Eddigi ++ [C], % Költséges művelet!
83     [{C, Eddigi} | utak(Bfa, Eddigil)] ++ utak(Jfa, Eddigil).
84
85 % 8. a)
86 cutak_a(C, Fa) ->
87     [CU || {C0,_} = CU <- utak(Fa), C0 == C].
88
89 % 8. b)
90 cutak_b(C, Fa) -> cutak(C, Fa, []).
91
92 % @spec ut(C::term(), F::fa(), Eddigi::ut()) ->
93 %     CimkezettUtak::[{C::term(), U::ut()}].
94 % A CimkezettUtak lista az F fa minden C címkéjű csomópontjához egy kételemű
95 % ennest társít, amelynek első eleme C, második eleme az Eddigi útvonal és
96 % és a csp. útvonala összefűzve.
97 cutak(_, level, _) ->
98     [];
99 cutak(C, {R,Bfa,Jfa}, Eddigi) ->
100     Eddigil = Eddigi ++ [R],
101     Cutak = cutak(C, Bfa, Eddigil) ++ cutak(C, Jfa, Eddigil),
102     if R == C -> [{C, Eddigi} | Cutak];
103     true -> Cutak
104     end.
105
106
107 %-----
108 %                               II. RÉSZ: LISTÁK
109 %-----
110
111
112 % 9.
113 zip([], []) ->
114     [];
115 zip([H1|T1], [H2|T2]) ->
116     [{H1,H2} | zip(T1,T2)].
117
118 unzip([]) ->
119     {[],[]};
120 unzip([H1,H2|T1]) ->
121     {T1,T2} = unzip(T),
122     {H1|T1}, [H2|T2]}.
123
124 % 10.
125 duplak([]) -> [];
126 duplak(L) ->
127     [ E || {E,E} <- zip(lists:sublist(L, length(L)-1), tl(L)) ].
128
129 % 11a.
130 all_different_a([]) ->
131     true;
132 all_different_a([H|T]) ->
133     not lists:member(H, T) andalso all_different_a(T).
134
135
136 % 11b.
137 all_different_b(L) -> length(L) == length(lists:usort(L)).
138
139 % 12.
140 desc() ->

```

```

141 Ds = lists:seq(1,6),
142 [ [A,B,C,D,E,F] || A <- Ds,
143 B <- Ds,
144 C <- Ds,
145 D <- Ds,
146 E <- Ds,
147 F <- Ds
148 ].
149
150 % 13. Nagyon lassú megoldás:
151 perms_0() ->
152 Ds = lists:seq(1,6),
153 [ [A,B,C,D,E,F] || A <- Ds,
154 B <- Ds,
155 C <- Ds,
156 D <- Ds,
157 E <- Ds,
158 F <- Ds,
159 % L <- [[A,B,C,D,E,F]] % generátorral tudnánk kötni vált
ozóhoz értéket, hogy ne kelljen 2x felépíteni L-et
160 all_different_b([A,B,C,D,E,F])
161 ].
162
163 % 13. Gyorsabb megoldás:
164 perms_1() ->
165 Ds = lists:seq(1,6),
166 [ [A,B,C,D,E,F] || A <- Ds,
167 B <- Ds -- [A],
168 C <- Ds -- [A,B],
169 D <- Ds -- [A,B,C],
170 E <- Ds -- [A,B,C,D],
171 F <- Ds -- [A,B,C,D,E]
172 ].
173
174 % 13. Rugalmasabb megoldás:
175 perms_2() ->
176 perms_2(lists:seq(1,6)).
177
178 % perms_2(L) az L lista elemeinek permutációit tartalmazó lista.
179 % Hátulról épít.
180 perms_2([X]) ->
181 [[X]];
182 perms_2(L) ->
183 [ [H|T] || H <- L, T <- perms_2(L--[H]) ].
184
185 perms_3() -> perms_3(lists:seq(1,6), []).
186
187 % perms_3(L) az L lista elemeinek permutációit tartalmazó,
188 % Prefix mögé fűzött lista. Előlről épít.
189 perms_3([], Prefix) ->
190 [Prefix];
191 perms_3(L, Prefix) ->
192 [ T || H <- L, T <- perms_3(L--[H], Prefix ++ [H]) ].
193
194
195 test(fa) ->
196 T1 = {4,
197 {3,level,level},
198 {6,
199 {5,level,level},
200 {7,level,level}}},
201 T2 = {a,
202 {b, {x,level,level}, level},
203 {c,
204 level,
205 {d,
206 {x,{e,level,level},level},
207 {a, {x,level,level},{x,level,level}}}},
208 T3 = {4,
209 {3,

```

```

210 {2,level,level},
211 {1,level,level}},
212 {6,
213 {5,level,level},
214 {7,level,level}}},
215 % lists:foldl(fun erlang:'and'/2, true,
216 [fa_noveltje(T1) := {5,{4,level,level}},{7,{6,level,level}},{8,level,level
}}}),
217 fa_tukorkepe(T1) := {4,{6,{7,level,level}},{5,level,level}},{3,level,lev
el}},
218 fa_balerteke(T1) := {ok, 3},
219 fa_balerteke(level) := error,
220 fa_balerteke(T2) := {ok, x},
221 fa_jobberteke(T1) := {ok, 7},
222 rendezett_fa(T1) := true,
223 rendezett_fa(T2) := false,
224 rendezett_fa(T3) := false,
225 rendezett_fa(T1) := rendezett_fa_2(T1),
226 rendezett_fa(T2) := rendezett_fa_2(T2),
227 rendezett_fa(T3) := rendezett_fa_2(T3),
228 rendezett_fa(T1) := rendezett_fa_3(T1),
229 rendezett_fa(T2) := rendezett_fa_3(T2),
230 rendezett_fa(T3) := rendezett_fa_3(T3),
231 tartalmaz(x, T1) := false,
232 tartalmaz(x, T2) := true,
233 elofordul(x, T1) := 0,
234 elofordul(x, T2) := 4,
235 utak(T1) := [{4,[]},{3,[4]},{6,[4]},{5,[4,6]},{7,[4,6]}],
236 utak(T2) := [{a,[]},{b,[a]},{x,[a,b]},{c,[a]},{d,[a,c]},{x,[a,c,d]},{
237 {e,[a,c,d,x]},{a,[a,c,d]},{x,[a,c,d,a]},{x,[a,c,d,a]}],
238 cutak_a(x,T1) := [],
239 cutak_a(x,T2) := [{x,[a,b]},{x,[a,c,d]},{x,[a,c,d,a]},{x,[a,c,d,a]}],
240 cutak_a(x,T1) := cutak_b(x,T1),
241 cutak_a(x,T2) := cutak_b(x,T2)
242 ]
243 %
244 ;
245
246 test(lista) ->
247 [
248 zip([1,2,3], [a,b,c]) := [{1,a},{2,b},{3,c}],
249 unzip([1,a],[2,b],[3,c]) := [1,2,3], [a,b,c]],
250 duplak([1,2,3]) := [],
251 duplak([1,1,2,3,3,3]) := [1,3,3],
252 all_different_a([1,3,5]) := true,
253 all_different_a([1,3,1]) := false,
254 all_different_b([1,3,5]) := true,
255 all_different_b([1,3,1]) := false,
256 begin
257 _ = statistics(runtime),
258 io:format("Futasi idot merek...~n"),
259 DescsLength = length(descs()),
260 io:format("descs ideje: ~w ms~n", [element(2, statistics(runtime))])
261 ,
262 P0 = perms_0(),
263 io:format("perms_0 ideje: ~w ms~n", [element(2, statistics(runtime))
264 ]),
265 P1 = perms_1(),
266 io:format("perms_1 ideje: ~w ms~n", [element(2, statistics(runtime))
267 ]),
268 P2 = perms_2(),
269 io:format("perms_2 ideje: ~w ms~n", [element(2, statistics(runtime))
270 ]),
271 P3 = perms_3(),
272 io:format("perms_3 ideje: ~w ms~n", [element(2, statistics(runtime))
273 ]),
274 DescsLength := 6*6*6*6*6*6
275 andalso
276 length(lists:usort([ lists:usort(P) || P <- [P0,P1,P2,P3] ])) := 1
277 end

```

```

273     ].
274
275
276 %-----
277 %                KIEGÉSZÍTÉS: 4. feladat kicsit hatékonyabb változatai
278 %-----
279
280 % 4. hatékonyabban (bonyolult, ráadásul lehetne még javítani)
281 rendezett_fa_2(level) ->
282     true;
283 rendezett_fa_2({C,Bfa,Jfa}=F) ->
284     Also = case fa_balerteke(Bfa) of
285         {ok, AE} -> AE;
286         error -> C
287     end,
288     Felso = case fa_jobberteke(Jfa) of
289         {ok, FE} -> FE;
290         error -> C
291     end,
292     rendezett_fa_2_kozott(Also,Felso,F).
293
294 % rendezett_fa_2_kozott(Also::term(), Felso::term(), F::fa()) -> B::bool().
295 % B igaz, ha F rendezett ÉS minden C címkéjére Also =< C andalso C =< Felso,
296 % az = csak akkor megengedett, ha C balérték (Also) vagy jobbérték (Felso).
297 rendezett_fa_2_kozott(Also, Felso, {C,level,level}) ->
298     Also =< C andalso C =< Felso;
299 rendezett_fa_2_kozott(Also, Felso, {C,Bfa,level}) ->
300     Also < C andalso C =< Felso andalso
301         rendezett_fa_2_kozott(Also,C,Bfa);
302 rendezett_fa_2_kozott(Also, Felso, {C,level,Jfa}) ->
303     Also =< C andalso C < Felso andalso
304         rendezett_fa_2_kozott(C,Felso,Jfa);
305 rendezett_fa_2_kozott(Also, Felso, {C,Bfa,Jfa}) ->
306     Also < C andalso C < Felso andalso
307         rendezett_fa_2_kozott(Also,C,Bfa) andalso
308         rendezett_fa_2_kozott(C,Felso,Jfa).
309
310 % 4. még hatékonyabban (csak egyszer járja be).
311 rendezett_fa_3(Fa) ->
312     case mmrendezett(Fa) of
313         {_,_,R} -> R;
314         R -> R
315     end.
316
317 %% @spec mmrendezett(F::fa()) ->
318 %%     {Min::term(), Max::term(), Rendezett::bool()} | Rendezett::bool().
319 %% Az F fa balértéke Min, jobbértéke Max, és rendezettsége Rendezett,
320 %% de olykor Min és Max nem kerül meghatározásra;
321 %% ezen esetek: üres fa, bizonyos nem rendezett fák.
322 mmrendezett(level) ->
323     true;
324 mmrendezett({C,Bfa,Jfa}) ->
325     % Első case-ben a bal fa rendezettségét vizsgáljuk,
326     % ezután BR a bal fa rendezettsége, Min a balérték,
327     % CBR a jobb fa elhagyásával keletkező {C,Bfa,level} fa rendezettsége.
328     CBR = case mmrendezett(Bfa) of
329         {BMin,BMax,BR} -> Min=BMin,
330             BR andalso BMax < C; % CBR a case értéke
331     BR -> Min=C,
332         BR % CBR a case értéke
333     end,
334     %% Második case-ben a jobb fa rendezettségét vizsgáljuk, és azt, hogy a
335     %% teljes fa rendezett-e (tudjuk, hogy {C,Bfa,level} fa rendezett).
336     if CBR -> case mmrendezett(Jfa) of
337         {JMin,JMax,JR} -> {Min,JMax,JR andalso C<JMin}; % eredmény
338         JR -> {Min,C,JR} % eredmény
339     end;
340     true -> false % eredmény (CBR:=false, tehát nem rendezett)
341 end.
342

```