

```

1 -module('dp19a-gy5').
2 -compile(export_all).
3 -author('patai@iit.bme.hu, hanak@emt.bme.hu, kapolnai@iit.bme.hu').
4 -vsn('$LastChangedDate: 2019-11-05 09:17:30 +0100 (k, 05 nov 2019) $$').
5
6 %-----
7 %
8 %-----
9
10 -type fa()      :: level | {any(),fa(),fa()}.
11 -type egeszfa() :: level | {integer(),egeszfa(),egeszfa()}.
12
13
14 % 1.
15 -spec fa_noveltje(F0::egeszfa()) -> F::egeszfa().
16 % Az F fa minden címkéje eggyel nagyobb az F0 azonos helyen lévő címkéjénél.
17 fa_noveltje(level) ->
18     level;
19 fa_noveltje({C,Bfa,Jfa}) ->
20     {C+1, fa_noveltje(Bfa), fa_noveltje(Jfa)}.
21
22 % 2.
23 -spec fa_tukorkepe(F0::fa()) -> F::fa().
24 % F az F0 fa tükörképe.
25 fa_tukorkepe(level) ->
26     level;
27 fa_tukorkepe({C,Bfa,Jfa}) ->
28     {C, fa_tukorkepe(Jfa), fa_tukorkepe(Bfa)}.
29
30
31 % 3.
32 -spec inorder(F::fa()) -> L::list().
33 % L az F fa elemeinek a fa inorder bejárásával létrejövő listája.
34 inorder(level) ->
35     [];
36 inorder({C,Bfa,Jfa}) ->
37     inorder(Bfa) ++ [C|inorder(Jfa)].
38
39 -spec preorder(F::fa()) -> L::list().
40 % L az F fa elemeinek a fa preorder bejárásával létrejövő listája.
41 preorder(level) ->
42     [];
43 preorder({C,Bfa,Jfa}) ->
44     [C|preorder(Bfa) ++ preorder(Jfa)].
45
46 -spec postorder(F::fa()) -> L::list().
47 % L az F fa elemeinek a fa postorder bejárásával létrejövő listája.
48 postorder(level) ->
49     [];
50 postorder({C,Bfa,Jfa}) ->
51     postorder(Bfa) ++ (postorder(Jfa) ++ [C]).
52
53
54 % 4.
55 -spec tartalmaz(C::any(), F::fa()) -> B::boolean().
56 % B igaz, ha C az F fa valamely címkéje.
57 tartalmaz(_, level) ->
58     false;
59 tartalmaz(C, {C,_,_}) ->
60     true;
61 tartalmaz(C, {_,Bfa,Jfa}) ->
62     tartalmaz(C, Bfa) orelse tartalmaz(C, Jfa).
63
64

```

```

65 % 5.
66 -spec elofordul(C::any(), F::fa()) -> N::integer().
67 % A C címke az F fában N-szer fordul elő.
68 elofordul(_, level) ->
69     0;
70 elofordul(C, {R,Bfa,Jfa}) ->
71     if C == R -> 1;
72     true -> 0
73     end
74     + elofordul(C, Bfa) + elofordul(C, Jfa).
75
76
77 % 6.
78 -spec cimkek(F::fa()) -> L::[any()].
79 % L az F címkéinek listája inorder sorrendben.
80 cimkek(Fa) -> cimkek(Fa, []).
81
82 -spec cimkek(F::fa(), L0::[any()]) -> L::[any()].
83 % L az F címkéinek listája inorder sorrendben L0 elé fűzve.
84 cimkek(level, L) ->
85     L;
86 cimkek({R,Bfa,Jfa}, L) ->
87     cimkek(Bfa, [R|cimkek(Jfa, L)]).
88
89
90 % 7.
91 -spec fa_balerteke(F::fa()) -> {ok, C::any()} | error.
92 % Egy nemüres F fa bal oldali szélső címkéje C (minden
93 % felmenőjére is igaz, hogy bal oldali gyermek).
94 fa_balerteke(level) ->
95     error;
96 fa_balerteke({C,level,_}) ->
97     {ok, C};
98 fa_balerteke({_,Bfa,_}) ->
99     fa_balerteke(Bfa).
100
101
102 % 8.
103 -spec fa_jobberteke(F::fa()) -> {ok, C::any()} | error.
104 % Egy nemüres F fa jobb oldali szélső címkéje C (minden
105 % felmenőjére is igaz, hogy jobb oldali gyermek).
106 fa_jobberteke(level) ->
107     error;
108 fa_jobberteke({C,_,level}) ->
109     {ok, C};
110 fa_jobberteke({_,_,Jfa}) ->
111     fa_jobberteke(Jfa).
112
113
114 % 9. a)
115 -spec rendezett_fa(F::fa()) -> B::boolean().
116 % B igaz, ha az F fa rendezett.
117 rendezett_fa(F) ->
118     L = cimkek(F),
119     L == lists:sort(L).
120

```

```

121 % 9. b)
122 -spec rendezett_fa_2(F::fa()) -> B::boolean().
123 % B igaz, ha az F fa rendezett.
124 rendezett_fa_2(level) ->
125     true;
126 rendezett_fa_2({C,Bfa,Jfa}) ->
127     case fa_jobberteke(Bfa) of
128         error ->
129             true;
130         {ok,J} ->
131             J < C
132     end
133     andalso
134     rendezett_fa_2(Bfa)
135     andalso
136     case fa_balerteke(Jfa) of
137         error ->
138             true;
139         {ok,B} ->
140             C < B
141     end
142     andalso
143     rendezett_fa_2(Jfa).
144
145 % 10.
146 -type ut() :: [any()].
147 -spec utak(F::fa()) -> CimkezettUtak::[{C::any(), CU::ut()}].
148 % A CimkezettUtak lista az F fa minden csomópontjához egy {C,CU} párt
149 % társít, ahol C az adott csomópont címkéje, CU pedig az adott
150 % csomóponthoz vezető útvonal.
151 utak(Fa) -> utak(Fa, []).
152
153 -spec utak(F::fa(), Eddigi::ut()) -> CimkezettUtak::[{C::any(), U::ut()}].
154 % A CimkezettUtak lista az F fa minden csomópontjához egy {C,U} párt
155 % társít, ahol C az adott csomópont címkéje, U pedig az adott
156 % csomóponthoz vezető útvonal az Eddigi eddigi útvonal elé fűzve.
157 utak(level, _) ->
158     [];
159     utak({C,Bfa,Jfa}, Eddigi) ->
160         % Eddigil = Eddigi ++ [C], % Költséges művelet: O(n2)!
161         Eddigil = lists:reverse([C | lists:reverse(Eddigi)]), % Kevésbé: O(2n).
162         [{C, Eddigi} | utak(Bfa, Eddigil)] ++ utak(Jfa, Eddigil).
163
164 % 11. a)
165 -spec cutak(C::any(), F::fa()) -> Utak::[{C::any(), CU::ut()}].
166 % Utak azon csomópontok útvonalainak listája F-ben, amelyek címkéje C.
167 cutak(C, Fa) ->
168     [CU || {C0,_} = CU <- utak(Fa), C0 == C].
169
170 % 11. b)
171 -spec cutak_2(C::any(), F::fa()) -> Utak::[{C::any(), CU::ut()}].
172 % Utak azon csomópontok útvonalainak listája F-ben, amelyek címkéje C.
173 cutak_2(C, Fa) -> cutak_2(C, Fa, []).
174
175 -spec cutak_2(C::any(), F::fa(), Eddigi::ut()) ->
176     CimkezettUtak::[{C::any(), U::ut()}].
177 % A CimkezettUtak lista az F fa minden C címkéjű csomópontjához egy {C,U}
178 % párt társít, ahol C az adott csomópont címkéje, U pedig az adott
179 % csomóponthoz vezető útvonal az Eddigi eddigi útvonal elé fűzve.
180 cutak_2(_, level, _) ->
181     [];
182     cutak_2(C, {R,Bfa,Jfa}, Eddigi) ->
183         Eddigil = Eddigi ++ [R],
184         Cutak = cutak_2(C, Bfa, Eddigil) ++ cutak_2(C, Jfa, Eddigil),
185         if R == C -> [{C, Eddigi} | Cutak];
186         true -> Cutak
187     end.
188
189
190

```

```

191 %-----
192 %                                     LUSTA FARKÚ LISTA
193 %-----
194
195 -type lazy_list() :: nil | {any(), fun(() -> lazy_list())}.
196 % Ez a lista félig lusta: a fej mindig kiértékelődik, a farkok lusta.
197
198 -spec infseq(N::integer()) -> LL::lazy_list().
199 % Az N-nel kezdődő, egyesével növekedő egész számok lusta farkú listája LL.
200 infseq(N) -> {N, fun() -> infseq(N+1) end}.
201
202 % 12. a)
203 -spec nth(L::lazy_list(), N::integer()) -> X::any().
204 % Az L lusta lista N-edik eleme X (számozás 1-től).
205 nth({H,_T}, 1) ->
206     H;
207     nth({_H,T}, N) ->
208         nth(T(), N-1).
209
210 % 12. b)
211
212 -spec nth_2(L::lazy_list(), N::integer()) -> {ok, X::any()} | error.
213 % Az L lusta lista N-edik eleme X (számozás 1-től). A visszatérési
214 % érték az 'error' atom, ha az L lista üres, vagy ha N < 1.
215 nth_2({H,_T}, 1) ->
216     {ok, H};
217     nth_2({_H,T}, N) when N > 1 ->
218         nth_2(T(), N-1);
219     nth_2(_, _) ->
220         error.
221
222
223 % 13. a)
224 -spec fibs(Curr::integer(), Next::integer()) -> LL::lazy_list().
225 % A 0. és 1. Fibonacci-számokkal (Curr == 0, Next == 1) meghívva
226 % fibs-et, LL a Fibonacci-számokat tartalmazó lusta lista.
227 fibs(Curr, Next) ->
228     {Curr, fun() -> fibs(Next, Curr+Next) end}.
229
230 % 13. b)
231 -spec fib(N::integer()) -> F::integer().
232 % F az N-edik Fibonacci-szám.
233
234 fib(N) ->
235     nth(fibs(0,1), N+1).
236
237

```

```

238 %-----
239 %                               NZH Erlang (mintafeladatok)
240 %-----
241
242
243 % 14.
244 is_space(X) -> X == 32.
245
246 mitirki() ->
247   X1 = {{some,6}, [$A], length([fun erlang:'element'/2]),
248         (fun erlang:'>/2)(2,3)},
249   X2 = lists:map(fun is_space/1, lists:concat(["xx7", "a5", "8z"])),
250   F = fun lists:seq/2,
251   X3 = [F(X, Y) || X <- F(1, 2), Y <- lists:reverse(F(1, 3)), X <= Y],
252   [ X1 == {{some,6}, "A", 1, false},
253     X2 == [false, false, false, false, false, false, false],
254     X3 == [[1,2,3],[1,2],[1],[2,3],[2]]
255   ].
256
257
258 % 15. a)
259 -spec szigetek(Xs::[integer()]) -> Rss::[[integer()]].
260 % Az Xs pozitív elemekből álló, folytonos, maximális
261 % hosszúságú részlistáinak listája az Rss lista.
262
263 szigetek(Xs) ->
264   szigetek(lists:reverse(Xs), [], []).
265
266 szigetek([], [], Sss) ->
267   Sss;
268 szigetek([], Ss, Sss) ->
269   [Ss|Sss];
270 szigetek([X|Xs], Ss, Sss) when X > 0 ->
271   szigetek(Xs, [X|Ss], Sss);
272 szigetek([_X|Xs], [], Sss) ->
273   szigetek(Xs, [], Sss);
274 szigetek([_X|Xs], Ss, Sss) ->
275   szigetek(Xs, [], [Ss|Sss]).
276
277 -spec sziget_2(Xs::[integer()], Zs::[integer()]) ->
278   {Ss::[integer()], Ms::[integer()]}.
279 % Ss az Xs pozitív elemekből álló, folytonos, maximális
280 % hosszúságú első részlistája Zs elé fűzve, Ms pedig az Xs maradéka.
281 sziget_2([], Zs) ->
282   {Zs, []};
283 sziget_2([X|Xs], Zs) when X <= 0 ->
284   {Zs, Xs};
285 sziget_2([X|Xs], Zs) ->
286   sziget_2(Xs, [X|Zs]).
287
288 -spec szigetek_2(Xs::[integer()]) -> Rss::[[integer()]].
289 % Az Xs pozitív elemekből álló, folytonos, maximális
290 % hosszúságú részlistáinak listája az Rss lista.
291
292 szigetek_2([]) ->
293   [];
294 szigetek_2([X|Xs]) when X <= 0 ->
295   szigetek(Xs);
296 szigetek_2(Xs) ->
297   {Ss, Ms} = sziget_2(Xs, []),
298   [lists:reverse(Ss)|szigetek_2(Ms)].
299
300

```

```

301 % 15. b)
302 -spec szfold(Zs::[integer()]) -> Ys::[{Hossz::integer(),Csucs::integer()}].
303 % A negatív számokat nem tartalmazó Zs egészlistában előforduló szárazföldek
304 % hosszát és legmagasabb pontját leíró {Hossz, Csucs} párok listája Ys.
305
306 szfold(Zs) ->
307   F = fun (Xs) -> {length(Xs), lists:max(Xs)} end,
308   lists:map(F, szigetek(Zs)).
309
310 -spec szfold_2(Zs::[integer()]) -> Ys::[{Hossz::integer(),Csucs::integer()}].
311 % A negatív számokat nem tartalmazó Zs egészlistában előforduló szárazföldek
312 % hosszát és legmagasabb pontját leíró {Hossz, Csucs} párok listája Ys.
313
314 szfold_2(Zs) ->
315   F = fun (Xs) -> {length(Xs), lists:max(Xs)} end,
316   lists:map(F, szigetek_2(Zs)).
317
318
319 % 16. a)
320 -spec dec2hex(D::integer()) -> H::string().
321 % A D decimális szám $0-$9 és $A-$F karakterek füzéreként ábrázolt
322 % hexadecimális megfelelője H.
323
324 dec2hex(0) ->
325   [charcode(0)];
326 dec2hex(D) ->
327   dec2hex(D, []).
328
329 dec2hex(0, Hs) ->
330   Hs;
331 dec2hex(D, Hs) ->
332   dec2hex(D div 16, [charcode(D rem 16)|Hs]).
333
334 -spec charcode(C::integer()) -> H::char().
335 % A 0 <= C <= 15 egésznek megfelelő hexadecimális jegy a
336 % $0, ..., $9, $A, ..., $F tartományban.
337 charcode(C) when C < 10 ->
338   C+$0;
339 charcode(C) ->
340   C-10+$A.
341
342
343 % 16. b)
344 -type fa(Elem) :: e | {n, Elem, fa(Elem), fa(Elem)}.
345 -spec d2hfa(Fa0::fa(integer())) -> Fa::fa(string()).
346 % A Fa0 decimális számokat tartalmazó fa hexadecimális számokat
347 % tartalmazó megfelelője Fa.
348
349 d2hfa(e) ->
350   e;
351 d2hfa({n, D, Bfa, Jfa}) ->
352   {n, dec2hex(D), d2hfa(Bfa), d2hfa(Jfa)}.
353
354 %-----
355

```

```

356 test(def) ->
357   T0 = {1,
358         {2,
359           {4,level,level},
360           {5,level,level}
361         },
362         {3,level,level}
363       },
364   T1 = {4,
365         {3,level,level},
366         {6,
367           {5,level,level},
368           {7,level,level}
369         }
370       },
371   T2 = {a,
372         {b, {v,level,level}, level},
373         {c,
374           level,
375           {d,
376             {w, {x,level,level}, level},
377             {f, {x,level,level}, {y,level,level}
378           }
379         }
380       },
381   T3 = {4,
382         {3,
383           {2,level,level},
384           {1,level,level}
385         },
386         {6,
387           {5,level,level},
388           {7,level,level}
389         }
390       },
391   {T0,T1,T2,T3}
392 ;
393
394 test(fal) ->
395   {T0,T1,T2,_T3} = test(def),
396   [
397     fa_noveltje(T1) ==
398       {5, {4,level,level}, {7, {6,level,level}, {8,level,level}}},
399     fa_tukorkepe(T1) ==
400       {4, {6, {7,level,level}, {5,level,level}}, {3,level,level}},
401     inorder(T0) == [4,2,5,1,3],
402     preorder(T0) == [1,2,4,5,3],
403     postorder(T0) == [4,5,2,3,1],
404     inorder(T1) == [3,4,5,6,7],
405     preorder(T1) == [4,3,6,5,7],
406     postorder(T1) == [3,5,7,6,4],
407     tartalmaz(x, T1) == false,
408     tartalmaz(x, T2) == true,
409     elofordul(x, T1) == 0,
410     elofordul(x, T2) == 2
411   ]
412 ;

```

```

413 test(fa2) ->
414   {_T0,T1,T2,T3} = test(def),
415   [
416     cimkek(T1) == [3,4,5,6,7],
417     fa_balerteke(T1) == {ok, 3},
418     fa_balerteke(level) == error,
419     fa_balerteke(T2) == {ok, v},
420     fa_jobberteke(T1) == {ok, 7},
421     fa_jobberteke(T2) == {ok, y},
422     fa_jobberteke(level) == error,
423     rendezett_fa(T1) == true,
424     rendezett_fa(T2) == false,
425     rendezett_fa(T3) == false,
426     rendezett_fa_2(T1) == true,
427     rendezett_fa_2(T2) == false,
428     rendezett_fa_2(T3) == false,
429     utak(T1) == [[4,[]], {3,[4]}, {6,[4]}, {5,[4,6]}, {7,[4,6]}],
430     utak(T2) == [{a,[]}, {b,[a]}, {v,[a,b]}, {c,[a]}, {d,[a,c]}, {w,[a,c,d]},
431                 {x,[a,c,d,w]}, {f,[a,c,d]}, {x,[a,c,d,f]}, {y,[a,c,d,f]}],
432     cutak(x,T1) == [],
433     cutak(x,T2) == [{x,[a,c,d,w]}, {x,[a,c,d,f]}],
434     cutak_2(x,T1) == cutak(x,T1),
435     cutak_2(x,T2) == cutak(x,T2)
436   ]
437 ;
438 test(lusta) ->
439   [
440     nth(infseq(1),99) == 99,
441     nth_2(infseq(1),99) == {ok, 99},
442     nth_2(nil, 5) == error,
443     nth(fibs(0,1), 100) == 218922995834555169026,
444     nth_2(fibs(0,1), 100) == {ok,218922995834555169026},
445     nth_2(fibs(0,1), -1) == error,
446     fib(99) == 218922995834555169026
447   ]
448 ;
449 test(nzh) ->
450   [
451     szigetek([]) == [],
452     szigetek([-1,0,-2,-3,0,0]) == [],
453     szigetek([1,2,3,0,0,4,5,6]) == [[1,2,3], [4,5,6]],
454     szigetek_2([]) == [],
455     szigetek_2([-1,0,-2,-3,0,0]) == [],
456     szigetek_2([1,2,3,0,0,4,5,6]) == [[1,2,3], [4,5,6]],
457     szfold([0,1,0,3,4,0,0,1]) == [{1,1}, {2,4}, {1,1}],
458     szfold([0,0,30,4,10,0,0,100]) == [{3,30}, {1,100}],
459     szfold([0,0,0]) == [],
460     szfold_2([0,1,0,3,4,0,0,1]) == [{1,1}, {2,4}, {1,1}],
461     szfold_2([0,0,30,4,10,0,0,100]) == [{3,30}, {1,100}],
462     szfold_2([0,0,0]) == [],
463     dec2hex(0) == "0",
464     dec2hex(7) == "7",
465     dec2hex(14) == "E",
466     dec2hex(75) == "4B",
467     d2hfa(e) == e,
468     d2hfa({n, 14, e, e}) == {n, "E", e, e},
469     d2hfa({n, 75, {n,200,e,e}, {n,35,e,e}}) ==
470       {n, "4B", {n,"C8",e,e}, {n,"23",e,e}}
471   ]
472 .

```