

```

1 -module('dp19a-gyl').
2 -compile(export_all).
3 -author('patai@emt.bme.hu, hanak@emt.bme.hu, kapolnai@iit.bme.hu').
4 -vsn('$LastChangedDate: 2019-09-16 14:36:38 +0200 (h, 16 szept 2019) $$').
5
6 % 1.
7 -spec lnko(A :: integer(), B :: integer()) -> D :: integer().
8 %% A és B legnagyobb közös osztója D.
9 lnko(A, 0) ->
10   A;
11 lnko(A, B) ->
12   lnko(B, A rem B).
13
14 % 2.
15 -spec pi(I :: integer()) -> Pi :: float().
16 %% A pi, azaz a Ludolph-féle szám I-edik közelítő értéke Pi.
17 %% A Leibniz-féle sor: pi/4 = 1 - 1/3 + 1/5 - 1/7 + ...
18 pi(I) ->
19   4 * pi_4(I*2+1, 1, p, 0).
20
21 pi_4(N, K, p, Pi) when K =< N ->
22   pi_4(N, K+2, m, Pi+(1/K));
23 pi_4(N, K, m, Pi) when K =< N ->
24   pi_4(N, K+2, p, Pi-(1/K));
25 pi_4(N, K, _E, Pi) when K > N ->
26   Pi.
27
28 % 3.
29 -spec dec2rad(R :: integer(), I :: integer()) -> Ds :: [integer()].
30 %% Az I egész szám R alapú számrendszerbe konvertált, decimális számként
31 %% megadott számjegyeinek listája Ds.
32 dec2rad(R, I) ->
33   dec2rad(R, I, []).
34
35 dec2rad(_R, 0, Ds) ->
36   Ds;
37 dec2rad(R, I, Ds) ->
38   dec2rad(R, I div R, [I rem R | Ds]).
39
40 % 4.
41 -spec conv(R :: integer(), I :: integer()) -> {ok, Ds :: string()} | error.
42 %% Az I egész szám R alapú számrendszerbe konvertált jegyeinek listája Ds,
43 %% ha 2 <= R <=16, egyébként error.
44 conv(R, I) when R >= 2 andalso R <= 16 ->
45   {ok, [lists:nth(X+1,"0123456789abcdef") || X <- dec2rad(R, I) ]};
46 conv(_R, _I) ->
47   error.
48
49 % 5.
50 -spec safe_last(Xs::[any()]) -> {ok, X::any()} | error.
51 %% Ha Xs üres, akkor 'error', különben X az Xs lista utolsó eleme.
52 safe_last([_] = Xs) -> {ok, lists:last(Xs)};
53 safe_last(_) -> error.
54
55 % 6.
56 -spec nth(N::integer(), L::[any()]) -> E::any().
57 %% Az L lista N-edik eleme E (1-től számozva az elemeket).
58 nth(1, [_]) -> H;
59 nth(N, [_|T]) -> nth(N-1, T).
60
61 % 7.
62 -spec split(N::integer(), L::[any()]) -> {P::[any()], S::[any()]}.
63 %% A P lista az L lista N hosszú prefixuma (első N eleme), az S lista az
64 %% L lista (length(L) - N) hosszú szuffixuma (első N eleme utáni része).
65 split(0, L) ->
66   {[], L};
67 split(N, [_|T]) ->
68   {PT,S} = split(N-1, T),
69   {[H|PT], S}.
70

```

```

71 % 8.
72 -spec take(L0::[any()], N::integer()) -> L::[any()].
73 %% Az L0 lista N hosszú prefixuma az L lista.
74 take(L0, N) ->
75   {L1,_L2} = split(N, L0),
76   L1.
77
78 % vagy mintaillesztés nélkül BIF-fel:
79 -spec take_2(L0::[any()], N::integer()) -> L::[any()].
80 take_2(L0, N) ->
81   element(1, split(N, L0)).
82
83 % 9.
84 -spec drop(L0::[any()], N::integer()) -> L::[any()].
85 %% Az L0 lista első N elemét nem tartalmazó szuffixuma az L lista.
86 drop(L0, N) ->
87   {_L1,L2} = split(N, L0),
88   L2.
89
90 % vagy mintaillesztés nélkül BIF-fel:
91 -spec drop_2(L0::[any()], N::integer()) -> L::[any()].
92 drop_2(L0, N) ->
93   element(2, split(N, L0)).
94
95 % 10.
96 -spec tails(Xs::[any()]) -> Zss::[[any()]].
97 %% Az Xs lista egyre rövidülő szuffixumainak listája a Zss lista.
98 tails([_H|T]=Xs) -> [Xs|tails(T)];
99 tails([]) -> [[]].
100
101 % 11.
102 -spec prefixes(Xs::[any()]) -> Zss::[[any()]].
103 %% Az Xs lista egyre hosszabbodó prefixumainak listája a Zss lista.
104 prefixes(Xs) ->
105   prefixes(Xs, 0).
106
107 -spec prefixes(Xs::[any()], N::integer()) -> Zss::[[any()]].
108 % Az Xs lista legalább N hosszú prefixumainak listája a Zss lista.
109 prefixes(Xs, N) ->
110   case length(Xs) >= N of
111     true -> [take(Xs, N)|prefixes(Xs, N+1)];
112     false -> []
113   end.
114
115 % vagy őrrrel:
116 -spec prefixes_2(Xs::[any()]) -> Zss::[[any()]].
117 %% Az Xs lista egyre hosszabbodó prefixumainak listája a Zss lista.
118 prefixes_2(Xs) ->
119   prefixes_2(Xs, 0).
120
121 -spec prefixes_2(Xs::[any()], N::integer()) -> Zss::[[any()]].
122 % Az Xs lista legalább N hosszú prefixumainak listája a Zss lista.
123 prefixes_2(Xs, N) when length(Xs) >= N ->
124   [take(Xs, N)|prefixes_2(Xs, N+1)];
125 prefixes_2(_Xs, _N) ->
126   [].
127

```

```

128 % 12.
129 -spec sublists(N::integer(), Xs::[any()]) ->
130     [{B::integer(), Ps::[any()], A::integer()}].
131 %% Az Xs lista egy olyan (folytonos) részlistája az N hosszú Ps lista,
132 %% amely előtt B és amely után A számú elem áll Xs-ben.
133 sublists(N, Xs) ->
134     sublists(N, 0, Xs).
135
136 -spec sublists(N::integer(), B0::integer(), Xs::[any()]) ->
137     [{B::integer(), Ps::[any()], A::integer()}].
138 %% Az Xs lista egy olyan (folytonos) részlistája az N hosszú Ps lista,
139 %% amely előtt (B-B0) és amely után A számú elem áll Xs-ben.
140 sublists(N, B0, Xs) ->
141     case length(Xs) < N of
142     true -> [];
143     false ->
144         [{B0, take(Xs, N), length(Xs) - N} | sublists(N, B0+1, tl(Xs))]
145     end.
146
147 % 13.
148 -spec parban(Xs::[any()]) -> Zs::[any()].
149 %% A Zs lista az Xs lista összes olyan elemének listája, amelyet
150 %% vele azonos értékű elem követ.
151 parban([E,E|T]) -> [E|parban([E|T])];
152 parban([_E1,E2|T]) -> parban([E2|T]);
153 parban(_) -> [].
154
155 % 14.
156 -spec sublist(L0::[any()], S::integer(), N::integer()) -> L::[any()].
157 %% Az L0 lista S-edik elemétől kezdődő és N hosszú részlistája az L lista.
158 sublist([], _, _) -> [];
159 sublist([H|_], 1, 1) -> [H];
160 sublist([H|T], 1, N) -> [H|sublist(T, 1, N-1)];
161 sublist([_|T], S, N) -> sublist(T, S-1, N).
162
163 sublist_2(L, S, N) ->
164     take(drop(L, S-1), N).
165
166 sublist_3(L, S, N) ->
167     [lists:nth(I, L) | I <- lists:seq(S, S+N-1)].
168
169 % 15.
170 -spec vertekek(Xs::[any()]) -> Es::[any()].
171 %% Az Xs lista elemei közül a {v::atom(),E:any()} mintára illeszkedő
172 %% párok második tagjából képzett lista az Es lista.
173 vertekek(L) ->
174     [E | {v,E} <- L].
175
176 % vagy listanézet nélkül:
177 -spec vertekek_2(Xs::[any()]) -> Es::[any()].
178 vertekek_2(L) ->
179     lists:map(fun({v,E}) -> E end,
180             lists:filter(fun({v,_}) -> true ; (_,_) -> false end, L)).
181
182 % vagy mintaillesztés nélkül:
183 -spec vertekek_3(Xs::[any()]) -> Es::[any()].
184 vertekek_3(L) ->
185     lists:map(fun({v,E}) -> E end,
186             lists:filter(fun(X) -> is_tuple(X) andalso size(X) == 2 andalso
187                         element(1, X) == v end, L)).
188
189 % vagy a lista "darálásával"
190 -spec vertekek_4(Xs::[any()]) -> Es::[any()].
191 vertekek_4([_{v,E}|T]) ->
192     [E|vertekek_4(T)];
193 vertekek_4([_|T]) ->
194     vertekek_4(T);
195 vertekek_4([]) ->
196     [].
197

```

```

198 % 16.
199 -spec cons(X::any(), Xs::[any()]) -> Zs::[any()].
200 %% Az X elem Xs elé fűzésének az eredménye a Zs lista.
201 cons(X, Xs) ->
202     [X|Xs].
203
204 -spec append(Xs::[any()], Ys::[any()]) -> Zs::[any()].
205 %% Az Xs lista Ys elé fűzésének eredménye a Zs lista (Zs := Xs ++ Ys).
206 append(Xs, Ys) ->
207     lists:foldr(fun cons/2, Ys, Xs).
208
209 % 17.
210 -spec revapp(Xs::[any()], Ys::[any()]) -> Zs::[any()].
211 %% A megfordított Xs lista Ys elé fűzésének eredménye a Zs lista,
212 %% azaz Zs := lists:reverse(Xs)++Ys.
213 revapp(Xs, Ys) ->
214     lists:foldl(fun cons/2, Ys, Xs).
215
216 % 18.
217 -spec tails_2(Xs::[any()]) -> Zss::[[any()]].
218 %% Az Xs lista egyre rövidülő szuffixumainak listája a Zss lista.
219 tails_2(Xs) ->
220     lists:foldr(fun(Y, [Zs|Zss]) -> [[Y|Zs],Zs|Zss] end, [], Xs).
221
222 -spec tails_3(Xs::[any()]) -> Zss::[[any()]].
223 %% Az Xs lista egyre rövidülő szuffixumainak listája a Zss lista.
224 tails_3(Xs) ->
225     [drop(Xs, N) | N <- lists:seq(0, length(Xs))].
226
227 % 19.
228 -spec sublists_2(N::integer(), Xs::[any()]) ->
229     [{B::integer(), Ps::[any()], A::integer()}].
230 %% Az Xs lista egy olyan (folytonos) részlistája az N hosszú Ps lista,
231 %% amely előtt B és amely után A számú elem áll Xs-ben.
232 sublists_2(N, Xs) ->
233     [{B,take(drop(Xs, B), N),length(Xs)-B-N} |
234      B <- lists:seq(0, length(Xs)-N)].
235
236 % 20.
237 -spec parban_2(Xs::[any()]) -> Zs::[any()].
238 %% A Zs lista az Xs lista összes olyan elemének listája, amelyet
239 %% vele azonos értékű elem követ.
240 parban_2(Xs) ->
241     [E | [E,E|_] <- tails(Xs)].
242
243 -spec parban_3(Xs::[any()]) -> Zs::[any()].
244 %% A Zs lista az Xs lista összes olyan elemének listája, amelyet
245 %% vele azonos értékű elem követ.
246 parban_3(Xs) ->
247     [E | [E,E] <- lists:zip(tl(Xs), take(Xs, length(Xs)-1))].
248
249 % 21.
250 -spec dadogo(Xs::[any()]) -> Zss::[[any()]].
251 %% A Zss lista az Xs lista összes olyan nemüres (folytonos) részlistájából
252 %% álló lista, amelyet vele azonos értékű részlista követ.
253 dadogo(Xs) ->
254     [P | T <- tails(Xs),
255      N <- lists:seq(1, length(T) div 2),
256      begin {P,S} = split(N, T), P := take(S, N) end
257     ].
258

```

```

259 % 22.
260 -spec flatten(DeepList::list()) -> L::list(). %% list() := [any()].
261 %% A tetszőleges mélységű beágyazott listákból álló DeepList lista
262 %% elemeiből álló, listákat már nem tartalmazó lista az L lista.
263 flatten([]) ->
264   [];
265 flatten([H|T]) when is_list(H) ->
266   flatten(H) ++ flatten(T);
267 flatten([H|T]) -> % when not is_list(H)
268   [H|flatten(T)].
269
270 -spec flatten_2(DeepList::list()) -> L::list(). %% list() := [any()].
271 %% A tetszőleges mélységű beágyazott listákból álló DeepList lista
272 %% elemeiből álló, listákat már nem tartalmazó lista az L lista.
273 flatten_2(DL) -> flatten_2(DL, []).
274
275 -spec flatten_2(DeepList::list(), Tail::list()) -> L::list().
276 % flatten_2(DL, Tail) = Az L lista a kilapított s a Tail elé fűzött DL lista.
277 flatten_2([H|T], Tail) when is_list(H) ->
278   flatten_2(H, flatten_2(T, Tail));
279 flatten_2([H|T], Tail) ->
280   [H|flatten_2(T, Tail)];
281 flatten_2([], Tail) ->
282   Tail.
283
284 % Gyakorlo
285
286 -spec slash(F::fun((list()) -> {any(),list()}), Xs::(list())) ->
287   Zs::(list()).
288 %% A Zs lista az Xs listából az F ismételt alkalmazásával előálló lista, ahol
289 %% F eredménye egy pár: Zs következő eleme és Xs még fel nem dolgozott része.
290 slash(F, []) ->
291   [];
292 slash(F, Xs) ->
293   {Z, Rs} = F(Xs),
294   lists:append([Z], slash(F, Rs)).
295
296 -spec slash_2(F::fun((list()) -> {any(),list()}), Xs::(list()))
297   -> Zs::(list()).
298 %% A Zs lista az Xs listából az F ismételt alkalmazásával előálló lista, ahol
299 %% F eredménye egy pár: Zs következő eleme és Xs még fel nem dolgozott része.
300 slash_2(F, Xs) ->
301   slash_2(F, Xs, []).
302
303 slash_2(_F, [], Zs) ->
304   lists:reverse(Zs);
305 slash_2(F, Xs, Zs) ->
306   {Z, Rs} = F(Xs),
307   slash_2(F, Rs, [Z|Zs]).
308
309 rampa_1(L) ->
310   case L of
311     [X1,X2|Xs] -> case X1 =< X2 of
312       true -> {Zs,Ms} = rampa_1([X2|Xs]),
313         {[X1|Zs], Ms};
314       false -> {[X1], [X2|Xs]}
315     end;
316     L -> {L, []} % vagyis ha length(L) < 2
317   end.
318
319 % vagy örrel:
320 rampa_lb([]) -> % ha ures listaval hivjak meg
321   {[], []};
322 rampa_lb([X1,X2|Xs]) when X1 =< X2 ->
323   {Zs,Ms} = rampa_lb([X2|Xs]),
324   {[X1|Zs], Ms};
325 rampa_lb([X|Xs]) ->
326   {[X], Xs}.
327
328 % lásd http://www.erlang.org/doc/man/lists.html

```

```

329 rampa_2([]) ->
330   {[], []};
331 rampa_2(Ls) ->
332   Rs = lists:takewhile(fun({X,Y}) -> X =< Y end,
333     lists:zip(lists:sublist(Ls, length(Ls)-1), tl(Ls))),
334   lists:split(length(Rs)+1, Ls).
335
336 %% Alternatív megoldás tails/1 és lists:splitwith/2 használatával.
337 %%
338 %% rampa_3/1-ben tails/1 eredménye az Ls egyre rövidülő farkainak a listája,
339 %% pl.
340 %% tails([a,b,c,b,a]) := [a,b,c,b,a]
341 %%                               [b,c,b,a]
342 %%                               [c,b,a]
343 %%                               [b,a]
344 %%                               [a]
345 %%                               []
346 %% splitwith/2 közülük azokat adja vissza Rss-ben, amelyeknek a feje nem
347 %% nagyobb a második elemüknél, Mss-ben pedig - az első kivételével - azokat,
348 %% amelyekre ez nem áll fenn, pl.
349 %% Rss := [[a,b,c,b,a],[b,c,b,a]]
350 %% Mss := [[b,a],[a],[ ]]
351 %%
352 %% Ha ezek után az Rss lista üres, akkor egyedül Ls feje "képez" monoton
353 %% növekvő sorozatot, az Ls farka pedig a maradék.
354 %%
355 %% Ha az Rss nem üres, akkor első részlistájának a fejéből és összes
356 %% részlistájának a második eleméből képezzük a monoton növekvő sorozatot
357 %% (azaz a futamot, az eredménypár első tagját), az Mss minden nemüres
358 %% részlistájának az első eleméből pedig a maradéklistát (az eredménypár
359 %% második tagját).
360 rampa_3([]) ->
361   {[], []};
362 rampa_3(Ls) ->
363   F = fun([X1,X2|_] -> X1 =< X2; (_) -> false end,
364     {Rss,[_|Mss]} = lists:splitwith(F, tails(Ls)),
365     case Rss of
366       [] ->
367         lists:split(1, Ls);
368       [Es|_] ->
369         % {futam (eredménypár 1. tagja),
370         % maradéklista (eredménypár 2. tagja)}
371         {[hd(Es)|X || [_,X|_] <- Rss], [X || [X|_] <- Mss]}
372     end.
373
374 % -----
375

```

```

376 t() ->
377 Xs = [1,2,2,3,2,4,5,6,6,6,7,6,8,2,3,3,4,5,6,0,6,5,4,3,2,1],
378 Zs = [1,2,2,3],
379 Ms = [2,4,5,6,6,6,7,6,8,2,3,3,4,5,6,0,6,5,4,3,2,1],
380 (lnko(96,42) == 6)
381 and (pi(1000000) == 3.1415936535887745)
382 and (dec2rad(2, 13) == [1,1,0,1])
383 and (dec2rad(17, 127) == [7,8])
384 and (conv(16, 127) == {ok, "7f"})
385 and (conv(17, 127) == error)
386 and (safe_last([5,1,2,8,7]) == {ok,7})
387 and (safe_last([]) == error)
388 and (nth(2,[a,b,c]) == b)
389 and (split(3, [a,b,c,d,e]) == {[a,b,c],[d,e]})
390 and (take([10,20,30,40,50], 3) == [10,20,30])
391 and (take_2([10,20,30,40,50], 3) == [10,20,30])
392 and (drop([10,20,30,40,50], 3) == [40,50])
393 and (drop_2([10,20,30,40,50], 3) == [40,50])
394 and (prefixes([a,b,c]) == [[], [a], [a,b], [a,b,c]])
395 and (prefixes_2([a,b,c]) == [[], [a], [a,b], [a,b,c]])
396 and (tails([1,4,2]) == [[1,4,2],[4,2],[2],[ ]])
397 and (sublists(1,[a,b,c]) == [{0,[a],2}, {1,[b],1}, {2,[c],0}])
398 and (sublists(2,[a,b,c]) == [{0,[a,b],1}, {1,[b,c],0}])
399 and (parban([a,a,a,2,3,3,a,2,b,b,4,4]) == [a,a,3,b,4])
400 and (sublist([1,2,3,4],2,2) == [2,3])
401 and (sublist([1,2,3,4],2,3) == [2,3,4])
402 and (sublist_2([1,2,3,4],2,2) == [2,3])
403 and (sublist_2([1,2,3,4],2,3) == [2,3,4])
404 and (sublist_3([1,2,3,4],2,2) == [2,3])
405 and (sublist_3([1,2,3,4],2,3) == [2,3,4])
406 and (vertekek([alma, {s,1}, {v,1}, 3, {v,2}]) == [1,2])
407 and (vertekek_2([alma, {s,1}, {v,1}, 3, {v,2}]) == [1,2])
408 and (vertekek_3([alma, {s,1}, {v,1}, 3, {v,2}]) == [1,2])
409 and (vertekek_4([alma, {s,1}, {v,1}, 3, {v,2}]) == [1,2])
410 and (append([a,b,c], [1,2,3]) == [a,b,c,1,2,3])
411 and (revapp([a,b,c], [1,2,3]) == [c,b,a,1,2,3])
412 and (tails_2([1,4,2]) == [[1,4,2],[4,2],[2],[ ]])
413 and (tails_3([1,4,2]) == [[1,4,2],[4,2],[2],[ ]])
414 and (sublists_2(1,[a,b,c]) == [{0,[a],2}, {1,[b],1}, {2,[c],0}])
415 and (sublists_2(2,[a,b,c]) == [{0,[a,b],1}, {1,[b,c],0}])
416 and (parban_2([a,a,a,2,3,3,a,2,b,b,4,4]) == [a,a,3,b,4])
417 and (parban_3([a,a,a,2,3,3,a,2,b,b,4,4]) == [a,a,3,b,4])
418 and (dadogo([a,a,a,2,3,3,a,b,b,b,b]) ==
419 [a],[a],[3],[b],[b,b],[b],[b,b],[b],[b])
420 and (flatten([1,[2,3],[[[[4]]],[5,[6]]]) == [1,2,3,4,5,6])
421 and (flatten_2([1,[2,a],[[[[4]]],[5,["ab"]]]) == [1,2,a,4,5,97,98])
422 and (slash(fun(Ss) -> lists:split(3, Ss) end, "jaaaj!!! nem jooo!")
423 == ["jaa","aj!","!! ","nem"," jo","oo!"])
424 and (slash_2(fun(Ss) -> lists:split(3, Ss) end, "jaaaj!!! nem jooo!")
425 == ["jaa","aj!","!! ","nem"," jo","oo!"])
426 and (rampa_1(Xs) == {Zs, Ms})
427 and (rampa_2(Xs) == {Zs, Ms})
428 and (rampa_3(Xs) == {Zs, Ms})
429 and true.

```