

```

1 -module(dp11a_gy3).
2 -compile(export_all).
3 -author('patai@iit.bme.hu, hanak@iit.bme.hu, kapolnai@iit.bme.hu').
4 -vsn('2011-10-14 ($LastChangedDate: 2011-10-16 18:12:55 +0200 (v, 16 okt 2011)
) $$').
5
6 % 1.
7
8 lista_noveltje_1([]) -> [];
9 lista_noveltje_1([H|T]) -> [H+1|lista_noveltje_1(T)].
10
11 % 2.
12
13 last_1([X]) -> X;
14 last_1([_|T]) -> last_1(T).
15
16 % 3.
17
18 safe_last([_|_] = Xs) -> {ok, last_1(Xs)};
19 safe_last(_) -> error.
20
21 % 4.
22
23 split(0, L) ->
24   {[], L};
25 split(N, [H|T]) ->
26   {PT,S} = split(N-1, T),
27   {[H|PT], S}.
28
29 % 5.
30
31 take(L0, N) ->
32   element(1, split(N, L0)).
33
34 % 6.
35
36 drop(L0, N) ->
37   element(2, split(N, L0)).
38
39 % 7.
40
41 prefixes_1(Xs) ->
42   prefixes(Xs, 0).
43
44 %% prefixes(Xs::[term()], N::integer()) -> Xss::[[term()]].
45 %% Xss az Xs listának összes, legalább N hosszú prefixuma.
46 prefixes(Xs, N) ->
47   case length(Xs) >= N of
48     true -> [take(Xs, N)|prefixes(Xs, N+1)];
49     false -> []
50   end.
51
52 % vagy őrrel:
53 prefixes_b(Xs, N) when length(Xs) >= N ->
54   [take(Xs, N)|prefixes_b(Xs, N+1)];
55 prefixes_b(_Xs, _N) ->
56   [].
57
58 % 8.
59
60 tails_1([_H|T]=Xs) -> [Xs|tails_1(T)];
61 tails_1([]) -> [{}].
62
63 % 9.
64
65 sublists(N, Xs) ->
66   sublists(N, 0, Xs).
67
68 sublists(N, B, Xs) ->
69   case length(Xs) < N of

```

```

70     true -> [];
71     false -> [{B, take(Xs, N), length(Xs) - N} | sublists(N, B+1, tl(Xs))]
72   ]
73   end.
74
75 % 10.
76
77 sublists_1(Xs) ->
78   sublists_atleast(1, Xs).
79
80 %% sublists_atleast(N::integer(), Xs::[term()]) -> Xss::[[term()]].
81 %% Xss az Xs legalább N hosszú részlistáinak listája.
82 sublists_atleast(N, Xs) ->
83   case length(Xs) < N of
84     true -> [];
85     false -> sublists(N, Xs) ++ sublists_atleast(N+1, Xs)
86   end.
87
88 % 11.
89
90 parban_1([E,E|T]) -> [E|parban_1([E|T])];
91 parban_1([_E1,E2|T]) -> parban_1([E2|T]);
92 parban_1(_) -> [].
93
94 % 12.
95
96 cons(H, T) ->
97   [H|T].
98
99 append(Xs, Ys) ->
100   lists:foldr(fun cons/2, Ys, Xs).
101
102 % 13.
103
104 revapp(Xs, Ys) ->
105   lists:foldl(fun cons/2, Ys, Xs).
106
107 % 14.
108
109 tails_2(Xs) ->
110   lists:foldr(fun(Y, [Zs|Zss]) -> [[Y|Zs],Zs|Zss] end, [{}], Xs).
111
112 % 15.
113
114 map(F, L) ->
115   [F(X) || X <- L].
116
117 filter(F, L) ->
118   [X || X <- L, F(X)].
119
120 % 16.
121
122 lista_noveltje_2(L0) ->
123   [X+1 || X <- L0].
124
125 % 17.
126
127 tails_3(Xs) ->
128   [drop(Xs, N) || N <- lists:seq(0, length(Xs))].
129
130 % 18.
131
132 sublists_2(Xs) ->
133   [{B,take(drop(Xs, B), N),length(Xs)-B-N} || N <- lists:seq(1, length(Xs))
134   ,
135   B <- lists:seq(0, length(Xs)-
136   N)].
137
138 % vagy:
139
140

```

```

137 sublists_2b(Xs) ->
138     lists:flatten([sublists(N, Xs) || N <- lists:seq(1, length(Xs))]).
139
140 % 19.
141
142 parban_2(Xs) ->
143     [E || [E,E|_] <- tails_1(Xs)].
144
145 % 20.
146
147 dadogo(Xs) ->
148     [P || T <- tails_1(Xs),
149      N <- lists:seq(1, length(T) div 2),
150      begin {P,S} = split(N, T), P := take(S, N) end
151     ].
152
153 % 21.
154
155 rampa_1(L) ->
156     case L of
157     [X1,X2|Xs] -> case X1 =< X2 of
158         true -> {Zs,Ms} = rampa_1([X2|Xs]),
159         {X1|Zs}, Ms};
160     false -> {[X1], [X2|Xs]}
161     end;
162     L -> {L, []} % vagyis ha length(L) < 2
163     end.
164
165 % vagy örrel:
166
167 rampa_lb([]) -> % ha üres listával hívják meg
168     {[], []};
169 rampa_lb([X1,X2|Xs]) when X1 =< X2 ->
170     {Zs,Ms} = rampa_lb([X2|Xs]),
171     {[X1|Zs], Ms};
172 rampa_lb([X|Xs]) ->
173     {[X], Xs}.
174
175 % lásd http://www.erlang.org/doc/man/lists.html
176
177 rampa_2([]) ->
178     {[], []];
179 rampa_2(Ls) ->
180     Rs = lists:takewhile(fun({X,Y}) -> X =< Y end,
181     lists:zip(lists:sublist(Ls, length(Ls)-1), tl(Ls))),
182     lists:split(length(Rs)+1, Ls).
183
184 %% Alternatív megoldás tails/1 és lists:splitwith/2 használatával.
185 %%
186 %% rampa_3/1-ben tails/1 eredménye az Ls egyre rövidülő farkainak a listája,
187 %% pl.
188 %% tails([a,b,c,b,a]) := [a,b,c,b,a]
189 %%                      [b,c,b,a]
190 %%                      [c,b,a]
191 %%                      [b,a]
192 %%                      [a]
193 %%                      []
194 %% splitwith/2 közülük azokat adja vissza Rss-ben, amelyeknek a feje nem nagy
195 %% obb
196 %% a második elemüknél, Mss-ben pedig - az első kivételével - azokat amelyekr
197 %% e
198 %% ez nem áll fenn, pl.
199 %% Rss := [[a,b,c,b,a],[b,c,b,a]]
200 %% Mss := [[b,a],[a],[[]]
201 %%
202 %% Ha ezek után az Rss lista üres, akkor egyedül Ls feje "képez" monoton növe
203 %% kvo
204 %% sorozatot, az Ls farka pedig a maradék.
205 %%
206 %% Ha az Rss nem üres, akkor első részlistájának a fejéből és összes

```

```

203 %% részlistájának a második eleméből képezzük a monoton növekvő sorozatot (az
204 %% az
205 %% a futamot, az eredménypár első tagját), az Mss minden nemüres részlistáján
206 %% ak
207 %% az első eleméből pedig a maradéklistát (az eredménypár második tagját).
208 rampa_3([]) ->
209     {[], []];
210 rampa_3(Ls) ->
211     F = fun([X1,X2|_] -> X1 =< X2; (_) -> false end,
212     {Rss,[_|Mss]} = lists:splitwith(F, tails_1(Ls)),
213     case Rss of
214     [] ->
215         lists:split(1, Ls);
216     [Es|_] ->
217         % {futam (eredménypár 1. tagja), maradéklista (eredménypár 2. tag
218         ja}
219         {[hd(Es)|[X || [_,X|_] <- Rss]], [X || [X|_] <- Mss]}
220     end.
221
222 % -----
223 t() ->
224     Paros = fun(X) -> X rem 2 := 0 end,
225     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],
226     Zs = [1,2,2,3],
227     Ms = [2,4,5,6,6,6,7,6,8,2,3,3,4,5,6,0,6,5,4,3,2,1],
228     (lista_noveltje_1([1,5,2]) := [2,6,3])
229     and (last_1([5,1,2,8,7]) := 7)
230     and (safe_last([5,1,2,8,7]) := {ok,7})
231     and (safe_last([]) := error)
232     and (split(3, [a,b,c,d,e]) := {[a,b,c],[d,e]})
233     and (take([10,20,30,40,50], 3) := [10,20,30])
234     and (drop([10,20,30,40,50], 3) := [40,50])
235     and (prefixes_1([a,b,c]) := [[], [a], [a,b], [a,b,c]])
236     and (tails_1([1,4,2]) := [[1,4,2],[4,2],[2],[[]]])
237     and (sublists(1,[a,b,c]) := [{0,[a],2}, {1,[b],1}, {2,[c],0}])
238     and (sublists(2,[a,b,c]) := [{0,[a,b],1}, {1,[b,c],0}])
239     and (sublists_1([a,b]) := [{0,[a],1}, {1,[b],0}, {0,[a,b],0}])
240     and (parban_1([a,a,a,2,3,3,a,2,b,b,4,4]) := [a,a,3,b,4])
241     and (append([a,b,c], [1,2,3]) := [a,b,c,1,2,3])
242     and (revapp([a,b,c], [1,2,3]) := [c,b,a,1,2,3])
243     and (tails_2([1,4,2]) := [[1,4,2],[4,2],[2],[[]]])
244     and (map(Paros, [1,2,3,4]) := [false,true,false,true])
245     and (filter(Paros, [1,2,3,4]) := [2,4])
246     and (lista_noveltje_2([1,5,2]) := [2,6,3])
247     and (tails_3([1,4,2]) := [[1,4,2],[4,2],[2],[[]]])
248     and (sublists_2([a,b]) := [{0,[a],1}, {1,[b],0}, {0,[a,b],0}])
249     and (sublists_2b([a,b]) := [{0,[a],1}, {1,[b],0}, {0,[a,b],0}])
250     and (parban_2([a,a,a,2,3,3,a,2,b,b,4,4]) := [a,a,3,b,4])
251     and (dadogo([a,a,a,2,3,3,a,b,b,b,b]) := [[a],[a],[3],[b],[b,b],[b],[
252     b]])
253     and (rampa_1(Xs) := {Zs, Ms})
254     and (rampa_2(Xs) := {Zs, Ms})
255     and (rampa_3(Xs) := {Zs, Ms})
256     and true.

```