

```

1 /* -*- c++ -*-
2 * $Date: 2012-09-15 17:55:31 +0200 (szo, 15 szept 2012) $
3 * Copyright (C) 2011-2012, BME Declarative Programming Course,
4 * Peter Szeredi - Peter Hanak - Richard Kapolnai
5 *
6 * 2012.09.14. Gyakorlat - megoldasok
7 */
8
9 #include "ceklah"
10
11 ////////////////////////////////////////////////////
12 // megoldasok
13
14 // 1.
15 int csupa01(const int N) {
16     if (N == 0) return true;
17     if (N % 10 >= 2) return false;
18     return csupa01(N/10);
19 }
20
21 // 2.
22 // osztok(N,D) az N szam D-nel nem kisebb osztoinak szama
23 int osztok(const int N, const int D) {
24     if (N < D) return 0;
25     const int Ez = N % D == 0;
26     return Ez + osztok(N, D+1);
27 }
28 int osztok(const int N) {
29     return osztok(N, 1);
30 }
31
32 // 3.
33 // A es B legnagyobb olyan kozos osztója, ami nem nagyobb D-nel
34 int lnko(const int A, const int B, const int D) {
35     if (A % D == 0)
36         if (B % D == 0)
37             return D;
38     if (A % D + B % D == 0)
39         return D;
40     return lnko(A, B, D-1);
41 }
42 int lnko(const int A, const int B) {
43     return lnko(A, B, A);
44 }
45
46 // 4.
47 int lnko2(const int A, const int B) {
48     if (B == 0) return A;
49     return lnko2(B, A % B);
50 }
51
52 // 5.
53 int length(const list L) {
54     if (L == nil) return 0;
55     return 1 + length(tl(L));
56 }
57
58 // 5.*
59 // listaHossza_i(L, H) az L lista hossza + H
60 int lengthi(const list L, const int H) {
61     if (L == nil) return H;
62     return lengthi(tl(L), H + 1);
63 }
64 int lengthi(const list L) {
65     return lengthi(L, 0);
66 }
67
68 // 6.
69 list lista_noveltje(const list L) {
70     if (L == nil) return nil;

```

```

71     return cons(hd(L) + 1, lista_noveltje(tl(L)));
72 }
73
74 // 7.
75 // L nem lehet ures!
76 int last(const list L) {
77     if (tl(L) == nil) return hd(L);
78     return last(tl(L));
79 }
80
81 // 8.
82 list insert_nth(const list L, const int N, const int E) {
83     if (N == 0) return cons(E, L);
84     return cons(hd(L), insert_nth(tl(L), N - 1, E));
85 }
86
87 // 9.
88 int nthl(const list L, const int N) {
89     if (N == 1) return hd(L);
90     return nthl(tl(L), N - 1);
91 }
92
93 // 10.
94 list take(const list L, const int H) {
95     if (H == 0) return nil;
96     return cons(hd(L), take(tl(L), H - 1));
97 }
98
99 // 11.
100 list drop(const list L, const int B) {
101     if (B == 0) return L;
102     return drop(tl(L), B - 1);
103 }
104
105 // 12.
106 list sublist(const list L, const int H, const int B) {
107     return take(drop(L, B), H);
108 }
109
110 // 13.
111 list parban(const list L) {
112     if (L == nil) return nil;
113     if (tl(L) == nil) return nil;
114     if (hd(L) == hd(tl(L))) return cons(hd(L), parban(tl(L)));
115     return parban(tl(L));
116 }
117
118 // 13.*
119 list parbani(const list L, const list P) {
120     if (L == nil) return P;
121     if (tl(L) == nil) return P;
122     if (hd(L) == hd(tl(L))) return parbani(tl(L), cons(hd(L), P));
123     return parbani(tl(L), P);
124 }
125 list parbani(const list L) {
126     return parbani(L, nil);
127 }
128
129 // 14.
130 int inc(const int X) {
131     return X + 1;
132 }
133 list map(const funl F, const list L) {
134     if (L == nil) return nil;
135     return cons(F(hd(L)), map(F, tl(L)));
136 }
137 list lista_noveltje_2(const list L) {
138     return map(inc, L);
139 }
140

```

```

141 // 15.
142 int rdiv(const int E, const int A) {
143     return A / E;
144 }
145 int foldl(const fun2 F, const int A, const list L) {
146     if (L == nil) return A;
147     return foldl(F, F(hd(L), A), tl(L));
148 }
149
150 int plus1(const int A, const int B) {
151     return B + 1;
152 }
153 int length2(const list L) {
154     return foldl(plus1, 0, L);
155 }
156
157 // 16.
158 int first(const int B, const int J) {
159     return B;
160 }
161 int last2(const list L) {
162     return foldl(first, hd(L), L);
163 }
164
165 // 17.
166 int plus(const int A, const int B) {
167     return A + B;
168 }
169 int sum(const list L) {
170     return foldl(plus, 0, L);
171 }
172
173 // 18.
174 int sqplus(const int E, const int A) {
175     return E * E + A;
176 }
177 int sumsq(const list L) {
178     return foldl(sqplus, 0, L);
179 }
180
181 // 19.
182 int sq(const int X) {
183     return X * X;
184 }
185 int sumsq2(const list L) {
186     return sum(map(sq, L));
187 }
188
189 // 20.
190 // append
191 template <class Fun, class Elem>
192 Elem foldrt(const Fun F, const Elem A, const list L) {
193     if (L == nil) return A;
194     return F(hd(L), foldrt(F, A, tl(L)));
195 }
196
197 // L megforditva A elott
198 list append(const list L, const list A) {
199     if (L == nil) return A;
200     return cons(hd(L), append(tl(L), A));
201 }
202
203
204 ///////////////////////////////////////////////////
205 // megoldasok tesztje
206
207 int main() {
208     //writeln(l(10,20,30) == cons(10, cons(20, cons(30, nil))));
209     writeln(" I. Csupa 0 es/vagy 1 szamjegy");
210     writeln(csupa01(100) == true);

```

```

211     writeln(csupa01(2012) == false);
212     writeln(" 2. Osztok szama");
213     writeln(osztok(12) == 6);
214     writeln(" 3. Legnagyobb kozos oszto -- naiv megoldas");
215     writeln(lnko(6, 12) == 6);
216     writeln(lnko(6, 11) == 1);
217     writeln(lnko(6, 10) == 2);
218     writeln(" 4. Legnagyobb kozos oszto -- euklideszi algoritmus");
219     writeln(lnko2(6, 12) == 6);
220     writeln(lnko2(6, 11) == 1);
221     writeln(lnko2(6, 10) == 2);
222     writeln(" 5. Lista hossza");
223     writeln(length(l(1,3,5)) == 3);
224     writeln(" 5.* Lista hossza - jobbrekurziv változat");
225     writeln(lengthi(l(1,3,5)) == 3);
226     writeln(" 6. Szamlista minden elemenek novelese");
227     writeln(lista_noveltje(l(1,5,2)) == l(2,6,3));
228     writeln(" 7. Lista utolso elemenek meghatarozasa");
229     writeln(last(l(5,1,2,8,7)) == 7);
230     writeln(" 8. Beszuras listaba adott helyre");
231     writeln(insert_nth(l(1,8,3,5), 2, 6) == l(1,8,6,3,5));
232     writeln(insert_nth(l(1,3,8,5), 3, 3) == l(1,3,8,3,5));
233     writeln(" 9. Lista adott sorszamu eleme");
234     writeln(nthl(l(10,20,30), 3) == 30);
235     writeln(" 10. Lista adott hosszusagu prefixuma");
236     writeln(take(l(10,20,30,40,50), 3) == l(10,20,30));
237     writeln(" 11. Lista adott helyen kezdodo szuffixuma");
238     writeln(drop(l(10,20,30,40,50), 3) == l(40,50));
239     writeln(" 12. Reszlista kepzese");
240     writeln(sublist(l(10,20,30,40,50), 3, 1) == l(20,30,40));
241     writeln(" 13. Listaban parosaval elofordulo elemek listaja");
242     writeln(parban("aaabccabeedd") == "aaced");
243     writeln(" 13.* Listaban parosaval elofordulo elemek listaja - jobbrekurziv változat, ahol nem szamit az
eredmeny sorrendje");
244     writeln(parbani("aaabccabeedd") == "decaa");
245     writeln(" 14. A 6. feladat ujboli megoldasa a map függvennyel");
246     writeln(lista_noveltje_2(l(1,5,2)) == lista_noveltje(l(1,5,2)));
247     writeln(" 15. Az 5. feladat ujboli megoldasa a foldl függvennyel");
248     writeln(foldl(rdiv, 64, l(4,2,1)) == ((64/4)/2)/1); // vagyis 8.
249     writeln(length2(l(1,3,5)) == length(l(1,3,5)));
250     writeln(" 16. A 7. feladat ujboli megoldasa a foldl függvennyel");
251     writeln(last2(l(5,1,2,8,7)) == last(l(5,1,2,8,7)));
252     writeln(" 17. Lista elemeinek osszege foldl függvennyel");
253     writeln(sum(l(10,20,30)) == 60);
254     writeln(" 18. Lista elemeinek negyzetosszege a foldl függvennyel");
255     writeln(sumsq2(l(10,20,30)) == 1400);
256     writeln(" 19. A 18. feladat megoldasa a map es sum függvényekkel");
257     writeln(sumsq2(l(10,20,30)) == 1400);
258     writeln(" 20. ...melyikre emlekeztet a foldl függvény?");
259     const list L1 = "abc", L2 = "123";
260     writeln(foldrt(cons, L1, L2) == append(L2, L1));
261     writeln(append(L2, L1) == "123abc");
262 }

```